SPRIGHT: A Fast and Robust Framework for Sparse Walsh-Hadamard Transform

Xiao Li, Joseph K. Bradley, Sameer Pawar and Kannan Ramchandran*
Department of Electrical Engineering and Computer Science (EECS)
University of California, Berkeley
{xiaoli, josephkb, spawar, kannanr}@eecs.berkeley.edu

Abstract

We consider the problem of stably computing the Walsh-Hadamard Transform (WHT) of some N-length input vector in the presence of noise, where the N-point *Walsh spectrum* is K-sparse with $K = O(N^{\delta})$ scaling sub-linearly in the input dimension N for some $0 < \delta < 1$. Note that K is linear in N (i.e. $\delta = 1$), then similar to the standard Fast Fourier Transform (FFT) algorithm, the classic Fast WHT (FWHT) algorithm offers an $O(N)$ sample cost and $O(N \log N)$ computational cost, which are order optimal. Over the past decade, there has been a resurgence in research related to the computation of Discrete Fourier Transform (DFT) for some length-N input signal that has a K-sparse N-point *Fourier spectrum*. In particular, through a sparse-graph code design, our earlier work on the *Fast Fourier Aliasing-based Sparse Transform* (FFAST) algorithm [1] computes the K-sparse DFT in time $O(K \log K)$ by taking $O(K)$ noiseless samples. Inspired by the coding-theoretic design framework in [1], Scheibler et al. in [2] proposed the *Sparse Fast Hadamard Transform (SparseFHT)* algorithm that elegantly computes the K-sparse WHT in the *absence* of noise using $O(K \log N)$ samples in time $O(K \log^2 N)$. However, the SparseFHT algorithm explicitly exploits the noiseless nature of the problem, and is not equipped to deal with scenarios where the observations are corrupted by noise, as is true in general. Therefore, a question of critical interest is whether this coding-theoretic framework can be made robust to noise. Further, if the answer is yes, what is the extra price that needs to be paid for being robust to noise?

In this paper, we show, quite interestingly, that there is *no extra price* that needs to be paid for being robust to noise other than a constant factor. In other words, we can maintain the same scaling for the sample complexity $O(K \log N)$ and the computational complexity $O(K \log^2 N)$ as those of the noiseless case, using our proposed **SP***arse* **R***obust* **I***terative* **G***raph-based* **H***adamard* **T***ransform* (**SPRIGHT**) algorithm. Similar to the FFAST algorithm [1] and the SparseFHT algorithm [2], the proposed SPRIGHT framework succeeds with high probability with respect to a random ensemble of signals with sparse Walsh spectra, where the support of the non-zero WHT coefficients is uniformly random. Experiments further corroborate the robustness of the SPRIGHT framework as well as its scaling performance.

1 Introduction

Ever since the introduction of orthonormal Walsh functions, the Walsh-Hadamard Transform (WHT) has gained traction for signal analysis in place of the Discrete Fourier Transform (DFT) because of its simplicity in computations and applicability in the design of practical systems like digital circuits. Starting off as the "poor man's fast Fourier Transform", the WHT has been further deployed over the past few decades in image and video compression [3], spreading code design in multiuser systems such as CDMA and GPS [4], and compressive sensing [5]. More recently, sparsity in the *Walsh spectrum* is found in many real-world applications involving the processing of large datasets, such as learning (pseudo) Boolean functions, decision trees and disjunctive normative form (DNF) formulas, etc. Therefore, it is of practical and theoretical interest to develop fast algorithms for computing the

*This work was supported by grants NSF CCF EAGER 1439725, and NSF CCF 1116404 and MURI CHASE Grant No. 556016.

WHT of signals with sparse or approximately sparse Walsh spectra. Traditionally, the WHT can be computed using N samples and $O(N \log N)$ operations via a recursive algorithm [6,7] analogous to the Fast Fourier Transform (FFT). However, these costs can be significantly reduced if the signal has a sparse Walsh spectrum [8,9].

1.1 Motivation and Contributions

There has been a recent resurgence in research on computing the Discrete Fourier Transform (DFT) of signals that have sparse *Fourier spectra* [1,10–14]. Since the WHT is a special case of a multidimensional DFT over the binary field, recent advances in computing K-sparse N-point DFTs have provided insights in designing algorithms for computing sparse WHTs. In particular, major progress has been made in breaking the "N-barrier" for computing an N-point sparse DFTs, which means that the sample complexity and computational complexity do not depend on the signal dimension N. In particular, using a sparse-graph code design, the *Fast Fourier Aliasing-based Sparse Transform (FFAST)* algorithm [1] uses $O(K)$ samples and $O(K \log K)$ operations for any sub-linear sparsity $K = O(N^\delta)$ with $0 < \delta < 1$ assuming a uniform support distribution. Under a similar uniform support distribution for the WHT coefficients, the Sparse Fast Hadamard Transform (SparseFHT) algorithm developed in [2] elegantly computes a K-sparse N-point WHT with $K = O(N^\delta)$ using $O(K \log(N/K))$ samples and $O(K \log K \log N/K)$ operations by following the sparse-graph code design in [1] for DFTs. When K is scales sub-linearly in N as $K = O(N^\delta)$ for some constant $0 < \delta < 1$, these results are hereby interpreted as achieving a sample complexity $O(K \log N)$ and a computational complexity $O(K \log^2 N)$. A limitation of the SparseFHT algorithm is that it is designed to explicitly exploit the *noiseless* nature of the underlying signals and it is not clear how to generalize it to noisy settings. A key question of theoretical and practical interest in this paper is: what price must be paid to be robust to noise? Interestingly, in this paper we show that *there are no extra costs in sample complexity and computational complexity for being robust to noise, other than a constant factor determined by the signal-to-noise ratio (SNR).*

Inspired by the algorithm design from the FFAST algorithm in [1] and the noisy FFAST analysis in [15], we consider the problem of computing a K-sparse N-point WHT from the input vector *in the presence of noise*, when the sparsity $K = O(N^\delta)$ is sub-linear in the signal dimension N for some $0 < \delta < 1$ assuming a uniform support distribution. We develop a *SParse Robust Iterative Graph-based Transform (SPRIGHT)* framework to stably compute the K-sparse N-length WHT at any constant SNRs with high probability. In particular, our framework achieves sub-linear run-time $O(K \log^2 N)$ using $O(K \log N)$ noisy samples, which maintains the same sample and computational scaling as the noiseless case. This result also contrasts with the work on computing the sparse DFT in the presence of noise [15], where the robustness to noise incurs an extra factor of $O(\log N)$ in terms of the sample complexity from $O(K)$ to $O(K \log N)$ (the same extra factor is manifested in the run-time as well). This can be intuitively explained by the fact that the complex-valued N-point Fourier transform kernel has a "$1/N$ precision" while the binary-valued WHT kernel has a "bit precision".

1.2 Notation and Organization

Throughout this paper, the set of integers $\{0, 1, \cdots, N-1\}$ for some integer N is denoted by $[N]$. Lowercase letters, such as x, are used for the time domain expressions and uppercase letters, such as X, are used for the transform domain signal. Any boldface lowercase letter such as $\mathbf{x} \in \mathbb{R}^N$ represents a column vector containing the corresponding N samples. The operator $\mathrm{supp}(\mathbf{x})$ takes the support set of the vector $\mathbf{x}$ and $|\cdot|$ takes the cardinality of a certain set. The notation $\mathbb{F}_2$ refers to the finite field consisting of $\{0, 1\}$, with defined operations such as summation and multiplication modulo 2. Furthermore, we let $\mathbb{F}_2^n$ be the n-dimensional column vector with each element taking values from $\mathbb{F}_2$. For any vector $\mathbf{i} \in \mathbb{F}_2^n$, denote by $\mathbf{i} = [i[1], \cdots, i[n]]^T \in \mathbb{F}_2^n$ the index vector containing the binary representation of some integer i, with $i[1]$ and $i[n]$ being the least significant bit (LSB) and the most significant bit (MSB), respectively. The inner product of two binary indices $\mathbf{i} \in \mathbb{F}_2^n$ and $\mathbf{j} \in \mathbb{F}_2^n$ is defined by $\langle \mathbf{i}, \mathbf{j} \rangle = \sum_{t=0}^{n-1} i[t]j[t]$ with arithmetic over $\mathbb{F}_2$, and the inner product between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$ is defined

as $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{t=1}^{N} x[t]u[t]$ with arithmetic over $\mathbb{R}$. The sign function here is defined as

$$\mathsf{sgn}\,[x] = \begin{cases} 1, & x < 0 \\ 0, & x > 0 \end{cases} \tag{1}$$

such that $x = |x|(-1)^{\mathsf{sgn}[x]}$.

This paper is organized as follows. In Section 2, we present our input (signal) model and our goal, followed by a summary of our main results. To motivate our design, we explain in Section 3 the main idea of our SPRIGHT framework through a simple example. Then, we generalize the simple example and present the framework in Section 2, followed by detailed discussions in Section 5 about the noisy scenarios in our framework. Last but not least, in Section 6 we briefly mention some machine learning applications that can be potentially cast as a sparse WHT computation problem, followed by numerical experiments in Section 7.

2 Problem Setup and Main Results

Given a signal $\mathbf{x} \in \mathbb{R}^N$ containing $N = 2^n$ samples $x[\mathbf{m}]$ indexed by $\mathbf{m} \in \mathbb{F}_2^n$ (i.e. the n-bit binary representation of $m \in [N]$), its WHT coefficient is computed as

$$X[\mathbf{k}] = \frac{1}{\sqrt{N}} \sum_{\mathbf{m} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{k}, \mathbf{m} \rangle} x[\mathbf{m}], \tag{2}$$

where $\mathbf{k} = [k[1], \cdots, k[n]]^T \in \mathbb{F}_2^n$ denotes the n-tuple index in the transform domain. Likewise, each sample $x[\mathbf{m}]$ has a WHT expansion as

$$x[\mathbf{m}] = \frac{1}{\sqrt{N}} \sum_{\mathbf{k} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{m}, \mathbf{k} \rangle} X[\mathbf{k}]. \tag{3}$$

2.1 Problem Setup

In this work, we consider the noisy scenario where the samples $x[\mathbf{m}]$ are corrupted by additive noise $w[\mathbf{m}] \sim \mathcal{N}(0, \sigma^2)$, which is independent and normally distributed for all $\mathbf{m} \in \mathbb{F}_2^n$. Thus, we have access to only the noise-corrupted samples:

$$u[\mathbf{m}] = \frac{1}{\sqrt{N}} \sum_{\mathbf{k} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{m}, \mathbf{k} \rangle} X[\mathbf{k}] + w[\mathbf{m}], \quad \mathbf{m} \in \mathbb{F}_2^n. \tag{4}$$

Assumption 1. *Let $\mathbf{X} \in \mathbb{R}^N$ be the WHT coefficient vector with support $\mathcal{K} := \mathrm{supp}\,(\mathbf{X})$. Throughout this paper, we make the following assumptions:*

A1 *Each element in the support set $\mathcal{K}$ is chosen independently and uniformly at random from $[N]$.*

A2 *The sparsity $K = |\mathrm{supp}\,(\mathbf{X})| = O(N^\delta)$ is sub-linear in the dimension N for some $0 < \delta < 1$.*

A3 *Each coefficient $X[\mathbf{k}]$ for $\mathbf{k} \in \mathcal{K}$ is chosen from a finite set $\mathcal{X} := \{\pm\rho\}$ uniformly at random.*

A4 *The signal-to-noise ratio (SNR) is defined as*

$$\mathsf{SNR} = \frac{\|\mathbf{x}\|^2/N}{\sigma^2} = \frac{\rho^2}{\sigma^2 N/K} \tag{5}$$

and is assumed to be an arbitrary constant value (i.e., ρ scales with $\sqrt{N/K}$).

Remark 1. *While the uniform distribution assumption* **A1** *on the support $\mathcal{K}$ is essential to the analysis of our algorithm (see also [1] and [2]), it can be generalized to accommodate non-uniform distributions that are of practical interest in real world applications. If we fail to insist on the sub-linear sparsity regime imposed in* **A2***, our results reduce to $O(N)$ samples in time $O(N \log N)$, which is well understood in classic WHT computations. Further, the binary constellation assumption* **A3** *is imposed to simplify our analysis and can be readily extended to any arbitrarily large but finite constellation, which subsumes all practical digital signals that have been quantized with finite precision (essentially any signal processed by a digital computer). Last but not least, the constant SNR assumption* **A4** *covers all regimes of interest.*

The goal of this paper is to develop a robust and efficient algorithm that reliably recovers *exactly* the entire support $\mathcal{K}$ of the sparse WHT of a signal as well as the associated non-zero coefficients $X[\mathbf{k}]$ for $\mathbf{k} \in \mathcal{K}$ in the presence of noise. The questions of interest are

1. How many noisy samples are needed to reliably recover the support of the sparse WHT?
2. Can we reduce the computational complexity of the sparse WHT over that of the conventional WHT algorithm, even in the presence of noise?

In the following, we first provide a summary of our main technical results, followed by a brief mention of previous work on computing sparse transforms.

2.2 Main Result

Our design is characterized by the triplet $(M, T, \mathbb{P}_F)$, where M is the *sample complexity*[1], T is the *computational complexity* in terms of arithmetic operations, and $\mathbb{P}_F$ is the *probability of failure* in recovering the exact support of the sparse WHT, given by

$$\mathbb{P}_F := \mathbb{E}\left[1\{\text{supp}\left(\widehat{\mathbf{X}}\right) \neq \text{supp}\left(\mathbf{X}\right)\}\right], \tag{6}$$

where $1\{\cdot\}$ is the indicator function and $\text{supp}\left(\cdot\right)$ represents the support of some vector and the expectation is obtained with respect to the randomization of our algorithm, the noise distribution as well as the random signal ensemble in Assumption 1.

Theorem 1. *Let Assumption 1 hold for the signal of interest $\mathbf{x}$ and its WHT vector $\mathbf{X}$. Then for any sparsity regime $K = O(N^\delta)$ with $0 < \delta < 1$, the* **SPRIGHT** *framework computes the K-sparse N-point WHT $\mathbf{X}$ with a vanishing failure probability $\mathbb{P}_F \to 0$ asymptotically in K and N using the following two algorithm options:*

- *the* **Sample Optimal (SO) SPRIGHT** *algorithm with a sample complexity of $M = O(K \log N)$ and a computational complexity of $T = O(K \log^2 N)$;*
- *the* **Near Sample Optimal (NSO) SPRIGHT** *algorithm with a sample complexity of $M = O(K \log^2 N)$ and a computational complexity of $T = O(K \log^3 N)$.*

Proof. See Appendix A. □

Remark 2. *Since we assume an arbitrarily large but finite constellation $\mathcal{X}$ for each non-zero coefficient, we show that the coefficients can in fact be recovered perfectly, even from the noisy measurements with high probability. The recovery algorithm is equally applicable to support recovery for signals with arbitrary coefficients over the real field, but the analysis becomes overly cumbersome without offering more insights to our design. Hence we do not pursue it in this paper.*

Remark 3. *Note that although the result in Theorem 1 is obtained with a randomized algorithm, our SPRIGHT framework also admits the option of using a deterministic algorithm by spending an extra factor of $O(\log N)$ in both sample complexity and computational complexity.*

[1]Note that the sample complexity is the number of raw samples needed as input for computations, as opposed to the measurement complexity in compressed sensing, where each measurement may potentially require all the samples from the input vector.

2.3 Related Work

Due to the similarities between the DFT and the WHT, we give a brief account of previous work on reducing the sample and computational complexity of obtaining a K-sparse N-point DFT. The most related research thread in the literature is the computation of sparse DFT using theoretical computer science techniques such as sketching and hashing (see [14, 16–19]). Most of these algorithms aim at minimizing the approximation error of the DFT coefficients using an ℓ_2-norm metric instead of exact support recovery (i.e., ℓ_0-norm).

Among these works, the most recent progress in this direction is the sFFT (Sparse FFT) algorithm developed in the series of papers [10–12]. Most of these algorithms are based on first isolating (i.e., hashing) the non-zero DFT coefficients into different bins, using specific filters or windows that have 'good' (concentrated) support in both time and frequency. The non-zero DFT coefficients are then recovered iteratively, one at a time. The filters or windows used for the binning operation are typically of length $O(K \log N)$. As a result, the sample complexity is typically $O(K \log N)$ or more, with potentially large big-Oh constants as demonstrated in [13]. Then, [12] further improved the 2-D DFT algorithm for the special case of $K = \sqrt{N}$, which reduces the sample complexity to $O(K)$ and the computational complexity to $O(K \log K)$, albeit with a constant failure probability that does not vanish as the signal dimension N grows. On this front, the deterministic algorithm in [14] is shown to guarantee zero errors but with complexities of $O(\text{poly}(K, \log N))$. More recently, [20] develops a deterministic algorithm for computing a sparse WHT in time $O(K^{1+\epsilon} \log^{O(1)} N)$ with an arbitrary constant $\epsilon > 0$.

One of the interesting recent advances in computing sparse DFTs is in the breaking of the "N-barrier", which means that the complexities no longer depend on the input dimension N. In particular, the FFAST algorithm [1] uses only $O(K)$ samples and $O(K \log K)$ operations for any sparsity regime $K = O(N^{\delta})$ and $\delta \in (0, 1)$. Similar to the spirit of compressed sensing in linearly combining sparse components (i.e., DFT coefficients), the FFAST algorithm judiciously chooses subsampling patterns to create spectral aliasing patterns to make them look like "good" (i.e., near-capacity achieving) *erasure-correcting codes* [21, 22]. The key insight is that we can effectively transform the sparse DFT computation problem into that of sparse-graph decoding to reconstruct the original "message" (i.e., sparse spectrum), which allows to use a simple peeling-based decoder with very low complexity. The success of the FFAST algorithm depends on the *single-ton test* to pinpoint frequency bins containing only one "erasure event" (unknown non-zero DFT coefficient). Given such a single-ton bin, the value and location of the coefficient can be obtained and then removed from other bins. This procedure iterates until no more single-ton bins are found. In the same spirit of [1], the SparseFHT algorithm in [2] elegantly computes a K-sparse WHT of $\mathbf{x}$ using $O(K \log N)$ samples and $O(K \log^2 N)$ operations.

3 Main Idea: A Simple Example

Since the sparsity is much smaller than the input dimension $K \ll N$, it is desirable if we can compute the WHT using very few samples $M \ll N$ without reading the *entire* signal. The most straightforward way to reduce the number of samples to process is to *subsample*. However, from a reconstruction perspective, it is generally disastrous to subsample since it creates aliasing in the spectral domain that mixes the WHT coefficients $X[\mathbf{k}]$.

The key idea of our SPRIGHT framework is to embrace (rather than avoid) the aliasing pattern as a form of "alias code", which is induced by the subsampling patterns guided by coding-theoretic designs, and more specifically, sparse-graph codes such as Low Density Parity Check (LDPC) codes. Then, our SPRIGHT framework exploits the aliasing pattern (alias code) to reconstruct the sparse Walsh spectrum in the presence of noise, by uncovering the sparse coefficients one-by-one iteratively in the spirit of decoding over noisy channels. While the design philosophy is similar to the FFAST algorithm in [1] and the SparseFHT algorithm in [2], our framework non-trivially generalizes this to the noisy scenario by robustifying the "alias code" for noisy decoding. Interestingly, we show that our framework can maintain the same scaling in both sample complexity and computational complexity as that in the noiseless case [2]. For completeness, we will repeat the noiseless design in the sequel, but using our setup and terminology.

3.1 Subsampling and Aliasing

Our observation model is based on using multiple *basic observation sets* formed by *randomized subsampling* and *tiny-sized WHTs*, where each set contains $B = 2^b$ (for some $b > 0$) samples obtained as:

- *Subsampling*: consider some integer $b < n$, the subsampling of noisy signal $u[\mathbf{m}]$ in (4) is performed by isolating a subset of $B = 2^b$ samples indexed by $\mathbf{m} = \mathbf{M}\boldsymbol{\ell} + \mathbf{d}$ for $\boldsymbol{\ell} \in \mathbb{F}_2^b$, where $\mathbf{M} \in \mathbb{F}_2^{n \times b}$ is some binary matrix and $\mathbf{d} \in \mathbb{F}_2^n$ is some random binary vector. In other words, after generating $\mathbf{M} \in \mathbb{F}_2^{n \times b}$ and $\mathbf{d} \in \mathbb{F}_2^n$, the subset of samples are selected by running the b-tuple $\boldsymbol{\ell}$ over $\mathbb{F}_2^b$.
- *B-point WHT*: a much smaller B-point WHT is performed over the samples $u[\mathbf{M}\boldsymbol{\ell} + \mathbf{d}]$ for $\boldsymbol{\ell} \in \mathbb{F}_2^b$. The subsampled signal has an aliased WHT spectrum readily obtained by a B-point WHT

$$U[\boldsymbol{j}] = \sum_{\boldsymbol{\ell} \in \mathbb{F}_2^b} u[\mathbf{M}\boldsymbol{\ell} + \mathbf{d}](-1)^{\langle \boldsymbol{j}, \boldsymbol{\ell} \rangle}, \quad \boldsymbol{j} \in \mathbb{F}_2^b. \tag{7}$$

Example 1. *We consider an example with $n = 4$ and sparsity $K = B = 2^b = 4$ (i.e. $b = 2$). For simplicity, we construct 2 sets of observations using*

$$\mathbf{M}_1 = [\mathbf{0}_{2\times 2}^T, \mathbf{I}_{2\times 2}^T]^T, \quad \mathbf{M}_2 = [\mathbf{I}_{2\times 2}^T, \mathbf{0}_{2\times 2}^T]^T. \tag{8}$$

We call each set of observations using a different subsampling pattern a subsampling group. With these patterns, we access the following samples in each group for $\boldsymbol{\ell} = [\ell_1, \ell_2]^T \in \mathbb{F}_2^2$

$$u[\mathbf{M}_1\boldsymbol{\ell}] = u[0\ 0\ \ell_1\ \ell_2] \Longrightarrow \begin{cases} u[0000] \\ u[0001] \\ u[0010] \\ u[0011] \end{cases}, \quad u[\mathbf{M}_2\boldsymbol{\ell}] = u[\ell_1\ \ell_2\ 0\ 0] \Longrightarrow \begin{cases} u[0000] \\ u[0100] \\ u[1000] \\ u[1100] \end{cases}.$$

After performing a 4-point WHT on each set of these samples, we have 2 sets of noisy observations:

$$\begin{aligned}
U_1[00] &= X[0000] + X[0100] + X[1000] + X[1100] \quad + \quad W_1[00] \\
U_1[01] &= X[0001] + X[0101] + X[1001] + X[1101] \quad + \quad W_1[01] \\
U_1[10] &= X[0010] + X[0110] + X[1010] + X[1110] \quad + \quad W_1[10] \\
U_1[11] &= X[0011] + X[0111] + X[1011] + X[1111] \quad + \quad W_1[11] \\
U_2[00] &= X[0000] + X[0001] + X[0010] + X[0011] \quad + \quad W_2[00] \\
U_2[01] &= X[0100] + X[0101] + X[0110] + X[0111] \quad + \quad W_2[01] \\
U_2[10] &= X[1000] + X[1001] + X[1010] + X[1011] \quad + \quad W_2[10] \\
U_2[11] &= X[1100] + X[1101] + X[1110] + X[1111] \quad + \quad W_2[11].
\end{aligned}$$

3.2 Computing Sparse WHT as Sparse-Graph Decoding

In the presence of noise, the coefficients $X[\mathbf{k}]$ should be intuitively obtained as the "least-squares" solution over the 2 sets of B observations in Example 1. However, the linear regression problem is underdetermined as we are given 8 equations with 16 unknowns. Fortunately, the coefficients are sparse, and this helps significantly. For simplicity, suppose that the 4 non-zero coefficients are $X[0100] = 2, X[0110] = 4, X[1010] = 1$ and $X[1111] = 1$. Now we have 8 equations with 4 unknowns (non-zero), but we do not know which unknowns are non-zero. Then, we have

$$\begin{aligned}
U_1[00] &= X[0100] + W_1[00], & U_2[00] &= W_2[00] \\
U_1[01] &= W_1[01], & U_2[01] &= X[0100] + X[0110] + W_2[01] \\
U_1[10] &= X[0110] + X[1010] + W_1[10], & U_2[10] &= X[1010] + W_2[10] \\
U_1[11] &= X[1111] + W_1[11], & U_2[11] &= X[1111] + W_2[11].
\end{aligned}$$

Now this problem seems quite a bit less daunting since the number of equations is more than the number of unknowns. The challenging part, however, is that *we do not know in advance which coefficients* $X[\mathbf{k}]$ *exist* in the equation since the sparse coefficients are randomly chosen over $\mathbf{k} \in \mathbb{F}_2^n$. Here, we illustrate the principle of our recovery algorithm through the same simple example by showing that the recovery is an instance of sparse-graph decoding with the help of an "oracle" (described later). Then in the next subsection, we will introduce how to get rid of the oracle.

3.2.1 Oracle-based Sparse-Graph Decoding

The relationship between the observations $\{U_i[\boldsymbol{j}]\}_{i=1,2}^{\boldsymbol{j}\in\mathbb{F}_2^b}$ and the unknown coefficients $X[\mathbf{k}]$ can be shown as a bipartite graph in Fig. 1, where the left nodes (unknown coefficients $X[\mathbf{k}]$) and right nodes (observations $\{U_i[\boldsymbol{j}]\}_{i=1,2}^{\boldsymbol{j}\in\mathbb{F}_2^b}$) are referred to as the *variable nodes* and *check nodes* respectively in the language of sparse-graph codes. Depending on the connectivity of the sparse bipartite graph, we categorize the observations into the following types:

1. *Zero-ton*: a check node is a zero-ton if it has no non-zero coefficients (e.g., the color *blue* in Fig. 1).
2. *Single-ton*: a check node is a single-ton if it involves only one non-zero coefficient (e.g., the color *yellow* in Fig. 1). Specifically, we refer to the index $\mathbf{k}$ and its associated value $X[\mathbf{k}]$ as the *index-value pair* $(\mathbf{k}, X[\mathbf{k}])$.
3. *Multi-ton*: a check node is a multi-ton if it contains more than one non-zero coefficient (e.g., the color *red* in Fig. 1).

To illustrate our reconstruction algorithm, we assume that there exists an "oracle" that informs the decoder exactly which check nodes are *single-tons*. Furthermore, the oracle further provides the index-value pair for that single-ton. In this example, the oracle informs the decoder that check nodes labeled $U_1[00]$, $U_1[11]$, $U_2[10]$ and $U_2[11]$ are single-tons with index-value pairs $(0100, X[0100])$, $(1111, X[1111])$, $(1010, X[1010])$ and $(1111, X[1111])$ respectively. Then the decoder can subtract their contributions from other check nodes, forming new single-tons. Therefore generally speaking, with the oracle information, the peeling decoder repeats the following steps:

Step (1) select all the edges in the bipartite graph with right degree 1 (identify single-ton bins);

Step (2) remove (peel off) these edges as well as the corresponding pair of variable and check nodes connected to these edges.

Step (3) remove (peel off) all other edges connected to the variable nodes that have been removed in *Step (2)*.

Step (4) subtract the contributions of the variable nodes from the check nodes whose edges have been removed in *Step (3)*.

Finally, decoding is successful if all the edges are removed from the graph together with all the unknown coefficients $X[\mathbf{k}]$ such that all the WHT coefficients are decoded.

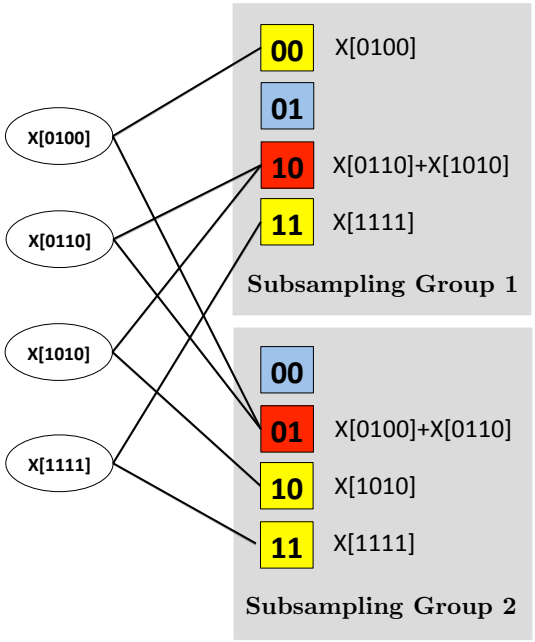


Figure 1: Example of a sparse bipartite graph consisting of 4 (non-zero) left nodes (variable nodes) connected to the 2 subsampling groups as a result of the sub-sampling-based randomized hashing in each group. Blue color represents "zero-ton", yellow color represents "single-ton" and red color represents "multi-ton".

3.2.2 Getting Rid of the Oracle : Bin Detection

Since the oracle information is critical in the peeling process, we proceed with our example and explain briefly how to obtain such information without an oracle. We call this procedure "bin detection". For simplicity, we illustrate the design where the samples are noise-free. To obtain the oracle information, we exploit the diversity of using different offsets. For instance, in group 1, we use the subsampling matrix $\mathbf{M}_1$ and the following set of offsets

$$\mathbf{d}_{1,0} = [0,0,0,0]^T, \quad \mathbf{d}_{1,1} = [1,0,0,0]^T, \quad \mathbf{d}_{1,2} = [0,1,0,0]^T, \quad \mathbf{d}_{1,3} = [0,0,1,0]^T, \quad \mathbf{d}_{1,4} = [0,0,0,1]^T.$$

In this way, using the subsampling pattern $\mathbf{M}_1$ and the offsets above, each check node is now assigned a 5-dimensional vector $\mathbf{U}_1[\boldsymbol{j}] = [U_{1,0}[\boldsymbol{j}], U_{1,1}[\boldsymbol{j}], U_{1,2}[\boldsymbol{j}], U_{1,3}[\boldsymbol{j}], U_{1,4}[\boldsymbol{j}]]^T$, where $U_{1,p}[\boldsymbol{j}]$ is associated with the p-th offset $\mathbf{d}_{1,p}$ for $p = 0, 1, \cdots, 4$. We call each vector of observations $\mathbf{U}_c[\boldsymbol{j}]$ in one group the *bin observation vector* $\boldsymbol{j}$. For example, the bin observation vectors for group 1 are obtained as $\mathbf{U}_1[00] = \mathbf{0}$ and

$$\mathbf{U}_1[01] = X[0100] \begin{bmatrix} 1 \\ (-1)^0 \\ (-1)^1 \\ (-1)^0 \\ (-1)^0 \end{bmatrix}, \ \mathbf{U}_1[10] = X[0110] \begin{bmatrix} 1 \\ (-1)^0 \\ (-1)^1 \\ (-1)^1 \\ (-1)^0 \end{bmatrix} + X[1010] \begin{bmatrix} 1 \\ (-1)^1 \\ (-1)^0 \\ (-1)^1 \\ (-1)^0 \end{bmatrix}, \ \mathbf{U}_1[11] = X[1111] \begin{bmatrix} 1 \\ (-1)^1 \\ (-1)^1 \\ (-1)^1 \\ (-1)^1 \end{bmatrix}.$$

Now with these bin observations, one can effectively determine if a check node is a zero-ton, a single-ton or a multi-ton. We go through some examples:

- *zero-ton bin*: consider the zero-ton check node $\mathbf{U}_1[00]$. A zero-ton check node can be identified easily since the measurements are all zero $\mathbf{U}_1[00] = \mathbf{0}$.
- *multi-ton bin*: consider the multi-ton check node $\mathbf{U}_1[10]$. A multi-ton can be easily identified since the magnitudes are not identical $|U_{1,0}[10]| \neq |U_{1,1}[10]| \neq |U_{1,2}[10]| \neq |U_{1,3}[10]| \neq |U_{1,4}[10]|$ or namely, the following ratio condition is not met:

$$\frac{U_{1,p}[10]}{U_{1,0}[10]} \neq \pm 1, \quad p = 1,2,3,4. \tag{9}$$

  Therefore, if the ratio test does not produce ± 1 or the magnitudes are not identical, we can conclude that this check node is a multi-ton.

- *single-ton bin*: consider the single-ton check node $\mathbf{U}_1[01]$. The underlying node is a single-ton if $|U_{1,0}[01]| = |U_{1,1}[01]| = |U_{1,2}[01]| = |U_{1,3}[01]| = |U_{1,4}[01]|$, or namely the ratio test produces all ± 1. Then, the index $\mathbf{k} = [k[1], k[2], k[3], k[4]]^T$ of a single-ton can be obtained by a simple ratio test

$$\begin{cases} (-1)^{\widehat{k}[1]} &= \dfrac{U_{1,1}[01]}{U_{1,0}[01]} = (-1)^0 \\ (-1)^{\widehat{k}[2]} &= \dfrac{U_{1,2}[01]}{U_{1,0}[01]} = (-1)^1 \\ (-1)^{\widehat{k}[3]} &= \dfrac{U_{1,3}[01]}{U_{1,0}[01]} = (-1)^0 \\ (-1)^{\widehat{k}[4]} &= \dfrac{U_{1,4}[01]}{U_{1,0}[01]} = (-1)^0 \end{cases} \Longrightarrow \begin{cases} \widehat{k}[1] = 0 \\ \widehat{k}[2] = 1 \\ \widehat{k}[3] = 0 \\ \widehat{k}[4] = 0 \\ \widehat{X}[\widehat{\mathbf{k}}] = U_{1,0}[01] \end{cases}$$

  Both the ratio test and the magnitude constraints are easy to verify for all check nodes such that the index-value pair is obtained for peeling.

This simple example shows how the problem of recovering the K-sparse coefficients $X[\mathbf{k}]$ can be cast as an instance of oracle-based peeling decoding by proper subsampling-induced *sparse bipartite graphs* in the dual domain. It further shows that the freedom in choosing offsets $\mathbf{d}$ gets rid of the oracle by *bin detection*. However, this simple example will not work in the presence of noise. The key idea of our design is that by carefully choosing the offsets $\mathbf{d}$ and subsampling patterns $\mathbf{M}$ through a sparse-graph coding lens, we can induce "peeling-friendly" sparse bipartite graphs that lead to fast recovery of the unknown WHT coefficients even in the presence of noise, as illustrated next.

4 The SPRIGHT Framework: General Architecture and Algorithm

In this section, we generalize the simple example and present the our proposed SPRIGHT framework. Our framework consists of an *observation generator* and a *reconstruction engine*, as shown in Fig. 2.

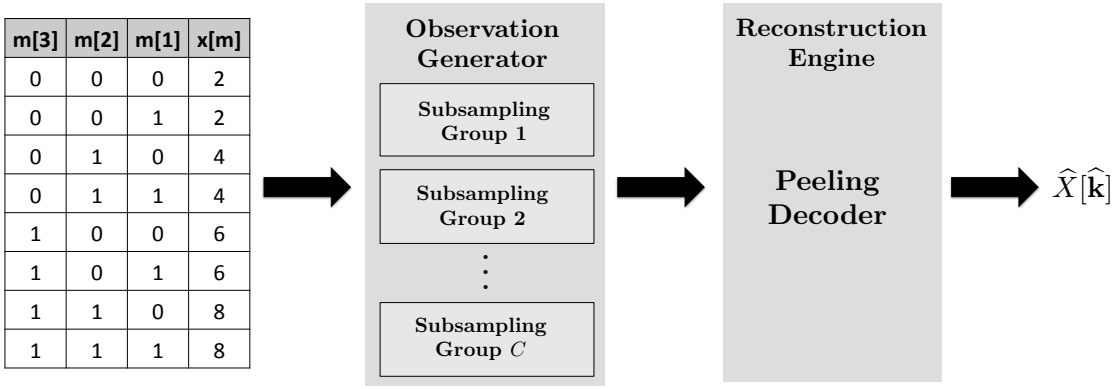


Figure 2: The conceptual diagram of our learning framework with C subsampling groups, where each group generates P basic query sets, each of size $B = 2^b$.

4.1 Observation Generator: Subsampling and Aliasing

In our SPRIGHT framework, the observations are obtained from C *subsampling groups*, where each group generates P *basic observation sets* of size $B = 2^b$. Each group uses a different matrix $\mathbf{M}_c \in \mathbb{F}_2^{n\times b}$ and a different set of P offsets $\mathbf{d}_{c,p} \in \mathbb{F}_2^n$ for $p \in [P]$, as summarized in Algorithm 1.

Algorithm 1 Subsampling and WHT

```
Input : u[m] for m ∈ F_2^n with N = 2^n;
Set : the number of subsampling groups C; observation set size B and number of observation sets P.
Generate : offsets d_{c,p} for p ∈ [P]; subsampling matrix M_c ∈ F_2^{n×b} for some b > 0
for c = 1 to C do
  for p = 1 to P do
    U_{c,p}[j] = sqrt(N/B) Σ_{ℓ∈F_2^b} u[M_c ℓ + d_{c,p}](−1)^{⟨j,ℓ⟩}.
  end for
end for
```

Proposition 1 (Basic Observation Model). *The B-point WHT coefficients indexed by $\boldsymbol{j} \in \mathbb{F}_2^b$ can be written as:*

$$U_{c,p}[\boldsymbol{j}] = \sum_{\mathbf{M}_c^T\mathbf{k}=\boldsymbol{j}} X[\mathbf{k}](-1)^{\langle \mathbf{d}_{c,p},\mathbf{k}\rangle} + W_{c,p}[\boldsymbol{j}], \quad p \in [P], \tag{10}$$

where $W_{c,p}[\boldsymbol{j}] = \sum_{\mathbf{M}_c^T\mathbf{k}=\boldsymbol{j}} W[\mathbf{k}](-1)^{\langle \mathbf{d}_{c,p},\mathbf{k}\rangle}$ and $W[\mathbf{k}]$ is the WHT coefficient of noise samples $w[\mathbf{m}]$.

Clearly, the $\boldsymbol{j}$-th WHT coefficient $U_{c,p}[\boldsymbol{j}]$ in each observation set is an aliased version (hash output) of the Walsh spectral coefficient $X[\mathbf{k}]$ under the hash function $\mathcal{H}_c : \mathbb{F}_2^n \to \mathbb{F}_2^b$ in the c-th group

$$\boldsymbol{j} = \mathcal{H}_c(\mathbf{k}) = \mathbf{M}_c^T\mathbf{k}, \quad c \in [C]. \tag{11}$$

It can be observed that the aliasing pattern (hash function) is invariant with respect to the offsets $\mathbf{d}_{c,p}$ used in subsampling. Similar to the *bin observation vector* in the simple example from Section 3.2.2, we can regroup the observations $U_{c,p}[\boldsymbol{j}]$ according to the hash $\mathcal{H}_c(\boldsymbol{j})$

$$\mathbf{U}_c[\boldsymbol{j}] \triangleq [\cdots, U_{c,p}[\boldsymbol{j}], \cdots]^T, \tag{12}$$

by stacking the $\boldsymbol{j}$-th WHT coefficient associated with all the offsets across the P observation sets in a vector.

Proposition 2 (Bin Observation Model). *Given the offset matrix $\mathbf{D}_c := [\cdots; \mathbf{d}_{c,p}; \cdots] \in \mathbb{F}_2^{P\times n}$, the $\boldsymbol{j}$-th bin observation vector in the c-th group can be written as*

$$\mathbf{U}_c[\boldsymbol{j}] = \sum_{\mathbf{k}:\ \mathbf{M}_c^T\mathbf{k}=\boldsymbol{j}} X[\mathbf{k}](-1)^{\mathbf{D}_c\mathbf{k}} + \mathbf{W}_c[\boldsymbol{j}], \quad \boldsymbol{j} \in \mathbb{F}_2^b,\ c \in [C], \tag{13}$$

where $(-1)^{(\cdot)}$ is the element-wise exponentiation operator and $\mathbf{W}_c[\boldsymbol{j}] = \sum_{\mathbf{M}_c^T\mathbf{k}=\boldsymbol{j}} W[\mathbf{k}](-1)^{\mathbf{D}_c\mathbf{k}}$ is the noise vector with $W[\mathbf{k}]$ being the WHT coefficient of the noise $w[\mathbf{m}]$.

Proof. The proof follows from WHT properties similar to that in [2], and hence is omitted here. □

From a coding-theoretic perspective, the observation vectors $\mathbf{U}_c[\boldsymbol{j}]$ for $\boldsymbol{j} \in \mathbb{F}_2^b$ across different groups $c \in [C]$ constitute the parity constraints of the coefficients $X[\mathbf{k}]$, where $X[\mathbf{k}]$ enters the $\boldsymbol{j}$-th parity of group c if $\mathbf{M}_c^T\mathbf{k} = \boldsymbol{j}$. It can be shown that if the set size $B = 2^b$ and the number of subsampling groups C are chosen properly, the bin observation vectors constitute parities of good error-correcting codes. Therefore, the coefficients can be uncovered iteratively in the spirit of peeling decoding (see Section 4.2), similar to that in LDPC codes. The key idea is to to avoid excessive aliasing by maintaining B on par with the sparsity $O(K)$ and imposing $P = O(\log N)$ for denoising purposes in bin detection. To keep our discussions focused, we defer the specific constructions of the subsampling model in terms of C, $B = 2^b$ and $\{\mathbf{M}_c\}_{c\in[C]}$ in Section 4.3.

4.2 Reconstruction Engine: Peeling Decoder

The outputs from the subsampling operation are then used for reconstruction. As stated in Proposition 2, each bin observation vector consists of linear combinations of the unknown WHT coefficients, which can be characterized by a *sparse bipartite graph* consisting of K left nodes (variable nodes) and CB right nodes (check nodes).

Definition 1 (Random Graph Ensemble). *For some redundancy parameter $\eta > 0$ let $B = \eta K = 2^b$ for some $b > 0$. The graph ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ consists of left C-regular sparse bipartite graphs where*

- *there are K left nodes (variable nodes), each labeled by a distinct element from the support $\mathbf{k} \in \mathcal{K}$;*
- *there are $B = 2^b$ right nodes (check nodes) per group, each labeled by the bin index $\boldsymbol{j} \in \mathbb{F}_2^b$ and assigned the bin observation vector $\mathbf{U}_c[\boldsymbol{j}]$;*
- *each left node $\mathbf{k}$ has degree C and each edge is connected to a right node $\boldsymbol{j}$ in each group according to the hash function $\mathcal{H}_c : \mathbb{F}_2^n \to \mathbb{F}_2^b$ given in* (11)*.*

Based on our simple example in Section 3.2, the unknown WHT coefficients (i.e. variable nodes) can be recovered through a peeling decoder over the graph ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$, as summarized in Algorithm 2. The key is to distinguish the observations $\mathbf{U}_c[\boldsymbol{j}]$ and identify single-ton bins for peeling.

In Algorithm 2, we denote the bin detection routine

$$\psi : \mathbb{R}^P \to (\texttt{type}, \widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}]) \tag{14}$$

which determines the types of bin observations:

1. $\mathbf{U}_c[\boldsymbol{j}]$ is a *zero-ton* if there does not exist $X[\mathbf{k}] \neq 0$ such that $\mathbf{M}_c^T\mathbf{k} = \boldsymbol{j}$, denoted by $\mathbf{U}_c[\boldsymbol{j}] \sim \mathcal{H}_{\mathrm{Z}}$;
2. $\mathbf{U}_c[\boldsymbol{j}]$ is a *single-ton* with the index-value pair $(\mathbf{k}, X[\mathbf{k}])$ if there exists only one $X[\mathbf{k}] \neq 0$ such that $\mathbf{M}_c^T\mathbf{k} = \boldsymbol{j}$, denoted by $\mathbf{U}_c[\boldsymbol{j}] \sim \mathcal{H}_{\mathrm{S}}(\mathbf{k}, X[\mathbf{k}])$;
3. $\mathbf{U}_c[\boldsymbol{j}]$ is a *multi-ton* if there exist more than one $X[\mathbf{k}] \neq 0$ such that $\mathbf{M}_c^T\mathbf{k} = \boldsymbol{j}$, denoted by $\mathbf{U}_c[\boldsymbol{j}] \sim \mathcal{H}_{\mathrm{M}}$.

Algorithm 2 Peeling Decoder

```
Input : observation vectors U_c[j] for j ∈ F_2^b, c ∈ [C];

Set : the number of peeling iterations I;
for i = 1 to I do
  for c = 1 to C do
    for j ∈ F_2^b do
      (type, k̂, X̂[k̂]) = ψ(U_c[j]).
      if type = single-ton then
        Peel off for all p = [P], c' = [C]
        Locate bin index j_{c'} = M_{c'}^T k̂
        U_{c',p}[j_{c'}] ← U_{c',p}[j_{c'}] − X̂[k̂](−1)^{⟨d_{c,p}, k̂⟩}.
      else if type ≠ single-ton then
        continue to next j.
      end if
    end for
  end for
end for
```

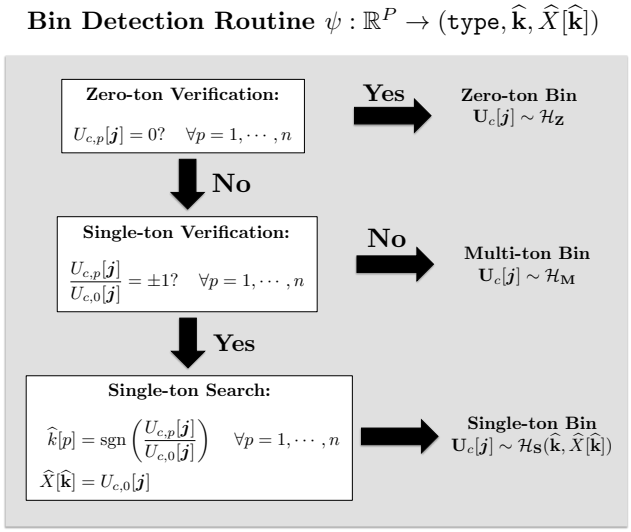


Figure 3: The bin detection routine $\psi : \mathbb{R}^P \to (\mathtt{type}, \widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}])$ for the noiseless setting by choosing offsets $\mathbf{D}_c = \mathbf{I}_{n\times n}$.

We focus on the noiseless case here (generalization of the simple example), and then elaborate on *robust bin detection* in the presence of noise in Section 5. The noiseless bin detection requires $P = n$ offsets through the steps summarized in Fig. 3:

- $\mathbf{U}_c[\boldsymbol{j}] \sim \mathcal{H}_{\mathrm{Z}}$ if $U_{c,p}[\boldsymbol{j}] = 0$ for all $p = 1, \cdots, n$.
- $\mathbf{U}_c[\boldsymbol{j}] \sim \mathcal{H}_{\mathrm{M}}$ if $|U_{c,p}[\boldsymbol{j}]/U_{c,0}[\boldsymbol{j}]| \neq \pm 1$ for all $p = 1, \cdots, n$.
- $\mathbf{U}_c[\boldsymbol{j}] \sim \mathcal{H}_{\mathrm{S}}(\mathbf{k}, X[\mathbf{k}])$ if the bin is neither a zero-ton nor a multi-ton.

The index-value pair $(\mathbf{k}, X[\mathbf{k}])$ of the single-ton is obtained as follows. Since each single-ton bin observation satisfies $U_{c,p}[\boldsymbol{j}] = X[\mathbf{k}](-1)^{\langle \mathbf{d}_{c,p}, \mathbf{k}\rangle}$, the corresponding sign[2] satisfies

$$\mathsf{sgn}\left[U_{c,p}[\boldsymbol{j}]\right] = \langle \mathbf{d}_{c,p}, \mathbf{k}\rangle \oplus \mathsf{sgn}\left[X[\mathbf{k}]\right], \tag{15}$$

where $\mathsf{sgn}\left[X[\mathbf{k}]\right]$ is the nuisance unknown sign. How do we get rid of such nuisance? This can be done by imposing a reference $\mathbf{d}_{c,0} = \mathbf{0}$ in addition to the offset matrix $\mathbf{D}_c \in \mathbb{F}_2^{P\times n}$ such that $\mathsf{sgn}\left[U_{c,0}\right] = \mathsf{sgn}\left[X[\mathbf{k}]\right]$.

This gives us a set of linear equations with respect to the unknown index $\mathbf{k}$:

$$\begin{bmatrix} \mathsf{sgn}\left[U_{c,1}[\boldsymbol{j}]\right] \oplus \mathsf{sgn}\left[U_{c,0}[\boldsymbol{j}]\right] \\ \mathsf{sgn}\left[U_{c,2}[\boldsymbol{j}]\right] \oplus \mathsf{sgn}\left[U_{c,0}[\boldsymbol{j}]\right] \\ \vdots \\ \mathsf{sgn}\left[U_{c,n}[\boldsymbol{j}]\right] \oplus \mathsf{sgn}\left[U_{c,0}[\boldsymbol{j}]\right] \end{bmatrix} = \mathbf{D}_c\mathbf{k}. \tag{16}$$

Clearly, if we choose the offsets in each group as $\mathbf{D}_c = \mathbf{I}_{n\times n}$, the unknown index $\mathbf{k}$ can be obtained directly from the signs of the observations. Finally, the value of the coefficient is obtained as $\widehat{X}[\widehat{\mathbf{k}}] = U_{c,0}[\boldsymbol{j}]$.

4.3 Subsampling Design and Algorithm Guarantees

With the general subsampling architecture given in Section 4.1, we discuss the specific constructions of the graph ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ by choosing appropriately the observation set size $B = 2^b$, the number of subsampling groups C, and the subsampling matrices $\{\mathbf{M}_c\}_{c\in[C]}$. We defer the discussion of how to choose offsets $\mathbf{d}_{c,p}$ to Section 5 because its design is independent of the graph ensemble.

[2]Note that the definition of the sign function here is a bit different than usual, where $\mathsf{sgn}\left[x\right] = 1$ if $x < 0$ and $\mathsf{sgn}\left[x\right] = 0$ if $x > 0$.

Let us first give some high level intuition of our subsampling design. Regardless of how many observation sets P are generated in each subsampling group $c \in [C]$, it is desirable to keep the number of subsampling groups C and the observation set size $B = 2^b$ small such that the resulting sample complexity is small. However, if C and B are too small, the resulting observation bins will end up mostly with multi-tons so the peeling operations get stuck. As a result, the subsampling design is about finding the "sweet spot" for the number of subsampling groups C and the observation set size B. In our analysis, we show that the product satisfies $CB = O(K)$, which implies that the subsampling using our generator does not introduce extra overheads other than a constant factor compared to the sparsity K. More importantly, from our analysis, such constant can be made explicit given the number of subsampling groups C.

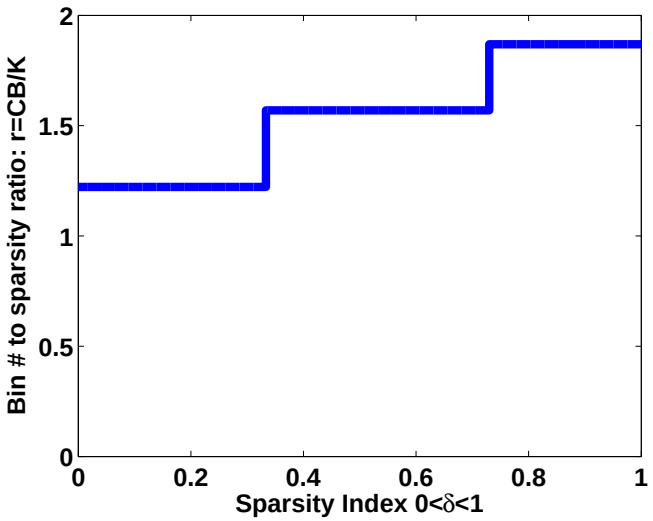


Figure 4: The ratio of the total bin number to the sparsity $r = CB/K$ as a function of the index $\delta \in (0, 0.99)$.

The subsampling design varies with the sparsity regime $0 < \delta < 1$ and hence, our results are stated with respect to different intervals of δ that cover the entire sparsity regime (see Appendix B). Our results stated below presents one constructive scheme using the partition[3] $(0, 1/3] \cup (1/3, 0.73] \cup (0.73, 7/8] \cup (7/8, 0.99]$. The sampling overhead (i.e. CB/K) introduced by the observation generator using this partition is shown in Fig. 4. This is by no means the unique scheme and the reason for choosing $1/3$, 0.73, $7/8$ and 0.99 as break points is that we want to keep the number of intervals small for the sake of presentation, since each interval results in a different design.

Theorem 2 (Oracle-based Peeling Decoder Performance). *Consider an input vector with a K-sparse WHT such that $K = O(N^\delta)$ for some $0 < \delta < 1$. Given an observation generator with C subsampling groups and an observation set size $B = \eta K$ for some $\eta > 0$, the subsampling-induced graph ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ guarantees that with probability at least $1 - O(1/K)$, the oracle-based peeling decoder recovers all K unknown coefficients in time $O(K)$ as long as*

- $C = 3$ *subsampling groups and* $B \geq 0.4073K$ *for* $0 < \delta \leq 1/3$ *(see Section B.3.1)*
- $C = 6$ *subsampling groups and* $B \geq 0.2616K$ *for* $1/3 < \delta \leq 0.73$ *(see Section B.3.2);*
- $C = 8$ *subsampling groups and* $B \geq 0.2336K$ *for* $0.73 < \delta \leq 0.875$ *(see Section B.3.3);*
- $C = 8$ *subsampling groups and* $B \geq 0.2336K$ *for* $0.875 < \delta \leq 0.99$ *(see Section B.3.4).*

Proof. Our analysis is similar to the arguments in [21, 22] using the so-called *density evolution* analysis from modern coding theory, which tracks the average density[4] of the remaining edges in the graph at each peeling iteration of the algorithm. Although the proof techniques are similar to those from [21] and [22], the graph used in our peeling decoder is different from those in [21, 22]. This leads to fairly important differences in the analysis, such as the degree distributions of the graphs and the expansion properties of the graphs (see Appendix B). Hence, we present an independent analysis here for our peeling decoder. In the following, we provide a brief outline of the proof elements highlighting the main technical components.

- *Density evolution* in Lemma 2: We analyze the performance of our peeling decoder over a *typical graph* (i.e., cycle-free) of the ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ for a fixed number of peeling iterations i. We

[3]We choose to cover the regime $0 < \delta \leq 0.99$ for the sake of presentation, and one can follow our proof in Appendix B to design subsampling patterns for $\delta > 0.99$.

[4]The density here refers to fraction of the remaining edges, or namely, the number of remaining edges divided by the total number of edges in the graph.

assume that a local neighborhood of every edge in the graph is cycle-free (tree-like) and derive a recursive equation that represents the average density of remaining edges in the graph at iteration i. The recursive equation guarantees that the average density is shrinking as the iterations proceed, as long as the redundancy parameter η is chosen accordingly with respect to the number of groups C for subsampling.

- *Convergence to density evolution* in Lemma 3: Using a Doob martingale argument [22] and [23], we show that the local neighborhood of most edges of a random graph from the ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ is cycle-free with high probability. This proves that with high probability, our peeling decoder removes all but an arbitrarily small fraction of the edges in the graph (i.e., the left nodes are removed at the same time after being decoded) in a constant number of iterations i.

- *Graph expansion property* for complete decoding in Lemma 4: We show that if the sub-graph consisting of the remaining edges is an "expander" (as will be defined later in this section), and if our peeling decoder successfully removes all but a sufficiently small fraction of the left nodes from the graph, then it removes all the remaining edges of the graph successfully. As long as the number of subsampling groups C is large enough for a given sparsity δ, we show that our graph ensemble is an expander with high probability. This completes the decoding of all the non-zero WHT coefficients.

□

5 Robust Bin Detection

We have shown in Section 4.3 that given an oracle for bin detection, our subsampling design for any sparsity regime $0 < \delta < 1$ guarantees that peeling decoder successfully recovers all unknown WHT coefficients in the absence of noise. In the noisy scenario, it is critical to robustify the *bin detection* scheme by choosing subsampling offsets differently than the noiseless setting. In the following, we explain the robust bin detection routine ψ. For simplicity, we drop the group index c and bin index $\boldsymbol{j}$ when we mention some bin observation. For example, the observation vector of some bin $\boldsymbol{j}$ from group c is denoted by $\boldsymbol{U} = [\cdots, U_p, \cdots]^T$, where the associated set of offsets is $\mathbf{D} = [\mathbf{d}_1; \cdots; \mathbf{d}_P] \in \mathbb{F}_2^{P\times n}$.

5.1 Performance Guarantees of Robust Bin Detection

From the noiseless design given in Section 4.2, we can see that the offset signature $(-1)^{\mathbf{Dk}}$ associated with each coefficient in Proposition 2 is the key to decode the unknown index-value pair $(\mathbf{k}, X[\mathbf{k}])$ of a single-ton. Let $\mathbf{S} = [\cdots, \mathbf{s}_\mathbf{k}, \cdots]$, where for each $\mathbf{k} \in \mathbb{F}_2^n$ we denote by

$$\mathbf{s}_\mathbf{k} = (-1)^{\mathbf{Dk}} \tag{17}$$

the offset signature *codebook* associated with the offset matrix $\mathbf{D}$. Then in the presence of noise, the bin observation vector can be written as

$$\boldsymbol{U} = \mathbf{S}\boldsymbol{\alpha} + \boldsymbol{W} \tag{18}$$

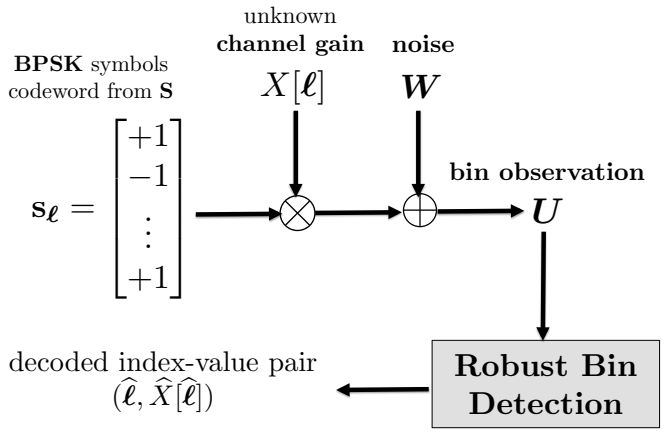


Figure 5: An illustration of a single-ton detection.

for some sparse vector $\boldsymbol{\alpha} = [\cdots, \alpha[\mathbf{k}], \cdots]^T$ such that $\alpha[\mathbf{k}] = X[\mathbf{k}]$ if $\mathbf{M}_c^T\mathbf{k} = \boldsymbol{j}$ and $\alpha[\mathbf{k}] = 0$ if otherwise. Clearly, the sparsity of $\boldsymbol{\alpha}$ implies the type of the bin. For example, the underlying bin is a single-ton if it is 1-sparse. It can be further shown from (2) that $\boldsymbol{W}$ follows a multivariate Gaussian distribution with zero mean and a covariance $\mathbb{E}\left[\boldsymbol{W}\boldsymbol{W}^T\right] = \nu^2\mathbf{I}$ and $\nu^2 := N\sigma^2/B$.

In the case of single-tons, the observation $\boldsymbol{U}$ can be regarded as the noise-corrupted version of some codeword from the codebook $\mathbf{S}$ (see Fig. 5). In our noiseless design, each codeword $\mathbf{s_k} \in \{-1, 1\}^n$ encodes the n-bit index $\mathbf{k}$ into n binary phase-shift keying (BPSK) symbols $(-1)^{\langle \mathbf{d}_p, \mathbf{k} \rangle} \in \{\pm 1\}$ for $p \in [n]$. This set of n BPSK symbols is scaled by the coefficient $X[\mathbf{k}]$ and observed as U_p for $p \in [n]$. This resembles the communication scenario where the goal of a receiver is to decode a sequence of n BPSK sequence with an unknown channel gain. Therefore, when there is additive noise in the channel, the codebook needs to be re-designed such that it can be robustly decoded.

In general, the vector $\boldsymbol{\alpha}$ is not necessarily 1-sparse (multi-ton bin). Through the *robust bin detection* scheme, we can effectively detect out the bins carrying some 1-sparse $\boldsymbol{\alpha}$ (i.e. single-tons), and recovers the index-value pair of the 1-sparse coefficient. Then, as the peeling operations proceed, the non-zero coefficients in other bins carrying $\boldsymbol{\alpha}$ that is not 1-sparse will be peeled off, which keeps forming new bins carrying 1-sparse vectors (single-ton).

In particular, we first present a straightforward design for *near-linear time detection* to shed some preliminary light on the noisy design, and then proceed to our proposed *sub-linear time detection* schemes. More specifically, we have two sub-linear time detection schemes that impose different sample complexities and computational complexities, called the *Sample-Optimal (SO)-SPRIGHT algorithm* and the *Near Sample-Optimal (NSO)-SPRIGHT algorithm* respectively.

Theorem 3. *Given the offsets $\mathbf{D} \in \mathbb{F}_2^{P \times n}$ chosen by*

- *Definition 2 for the near-linear time detection scheme, or*
- *Definition 3 for the NSO-SPRIGHT algorithm and Definition 4 for the SO-SPRIGHT algorithm,*

the failure probability $\mathbb{P}_F$ of the peeling decoder in the presence of noise is $O(1/K)$.

Proof. See Appendix D. □

5.2 Near-linear Time Robust Bin Detection: A Random Design

The near-linear time bin detection scheme follows the principle of using random codes to resolve the different bin hypotheses and obtain the index-value pair.

Definition 2. *Let $P = O(\log N)$. The near-linear time detection scheme requires P random offsets $\{\mathbf{d}_p\}_{p \in [P]}$ chosen independently and uniformly at random over $\mathbb{F}_2^n$ in every group.*

For some $\gamma \in (0, 1)$, the near-linear time detection routine is performed as follows:

- *zero-ton verification*: for zero-tons, we can expect the energy $\|\boldsymbol{U}\|^2$ to be small relative to the energy of a single-ton. Therefore, this idea is used to eliminate zero-tons:

$$\boldsymbol{U} \sim \mathcal{H}_Z, \quad \text{if } \frac{1}{P} \|\boldsymbol{U}\|^2 \leq (1+\gamma)\nu^2. \tag{19}$$

- *single-ton search*: after ruling out zero-tons and multi-tons, the ultimate goal is to identify single-tons in a certain group c in terms of the underlying index $\mathbf{k}$ and the value $X[\mathbf{k}]$ in that hash set $\{\mathbf{k} : \mathcal{H}_c(\mathbf{k}) = \boldsymbol{j}\}$. Therefore, assuming that the underlying bin $\boldsymbol{j}$ is a single-ton bin, we perform a single-ton search to estimate the pair of estimates $(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}])$ for peeling. To do so, we employ a Maximum Likelihood Estimate (MLE) test. For each of N/B possible coefficient locations $\mathbf{k}$ in $\mathbf{M}_c^T \mathbf{k} = \boldsymbol{j}$, we obtain the single-ton coefficient as

$$\widehat{\alpha}[\mathbf{k}] = \frac{1}{P} \mathbf{s}_\mathbf{k}^T \boldsymbol{U}, \quad \forall \mathbf{k} \text{ such that } \mathbf{M}_c^T \mathbf{k} = \boldsymbol{j}. \tag{20}$$

  Using the MLE of the coefficient, we choose among the locations by finding the location k which minimizes the residual energy:

$$\widehat{\mathbf{k}} = \arg\min_{\mathbf{k}} \ \|\boldsymbol{U} - \widehat{\alpha}[\mathbf{k}] \mathbf{s}_\mathbf{k}\|^2 . \tag{21}$$

With the estimated index $\widehat{\mathbf{k}}$, the value of the coefficient is obtained as

$$\widehat{X}[\widehat{\mathbf{k}}] = \begin{cases} \rho, & \text{if } \mathbf{s}_{\mathbf{k}}^T \boldsymbol{U}/P \geq 0 \\ -\rho, & \text{if } \mathbf{s}_{\mathbf{k}}^T \boldsymbol{U}/P < 0. \end{cases} \tag{22}$$

- *single-ton verification*: this step confirms if the bin is a single-ton via a residual test using the single-ton search estimates

$$\frac{1}{P} \left\| \boldsymbol{U} - \widehat{X}[\widehat{\mathbf{k}}] \mathbf{s}_{\widehat{\mathbf{k}}} \right\|^2 \leq (1+\gamma)\nu^2. \tag{23}$$

Since there are a total of ηK bins in each of the C subsampling groups and each bin has $P = O(\log N)$ measurements, the SPRIGHT framework using the near-linear time detection scheme leads to a sample cost of $M = C\eta KP = O(K \log N)$. In terms of complexity, solving the above minimizations requires an exhaustive search over all indices $\mathbf{M}_c^T k = \boldsymbol{j}$ for some bin $\boldsymbol{j} \in \mathbb{F}_2^b$. This leads to an exhaustive search over $O(N/K)$ elements on average in each peeling iteration, where each element imposes a search complexity of $P = O(\log N)$ by the generalized likelihood ratio test. As a result, across all $O(K)$ peeling iterations, this results in a total complexity of $T = O(N/K) \times O(\log N) \times O(K) = O(N \log N)$.

5.3 Sub-linear Time Robust Bin Detection

Inspired by the near-linear time bin detection scheme, we devise two simple schemes to achieve the same performance with sub-linear time complexity. Recall that the robust bin detection involves three steps:

1) zero-ton verification $\frac{1}{P} \|\boldsymbol{U}\|^2 \leq (1+\gamma)\nu^2$;

2) single-ton search that estimates the index-value pair $(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}])$;

3) single-ton verification $\frac{1}{P} \left\| \boldsymbol{U} - \widehat{X}[\widehat{\mathbf{k}}] \mathbf{s}_{\widehat{\mathbf{k}}} \right\|^2 \leq (1+\gamma)\nu^2$.

The near-linear time design is a straightforward construction of the offset matrix $\mathbf{D} \in \mathbb{F}_2^{P \times n}$ to guarantee success for step (1) and step (3). However, it does not optimize its choice of offsets to facilitate step (2) in the noisy setting, which causes the high complexity.

To avoid the joint estimation and detection approach in the near-linear time scheme, we use different offsets to tackle them separately. We perform the single-ton search using some offsets, while using other offsets for zero-ton and single-ton verifications. Since the fully random offsets already tackle the verifications with high probability, we can simply focus on designing offsets for the single-ton search. If the single-ton search can be performed with high probability of success using the same amount of samples and computations (in an order-sense), the entire bin detection scheme becomes sub-linear, as discussed in details below.

Proposition 3. *Given a single-ton bin with* $(\mathbf{k}, X[\mathbf{k}])$ *observed in noise*

$$U_p = X[\mathbf{k}](-1)^{\langle \mathbf{d}_p, \mathbf{k} \rangle} + W_p, \quad p \in [P], \tag{24}$$

the sign of each observation satisfies

$$\mathsf{sgn}\left[U_p\right] = \langle \mathbf{d}_p, \mathbf{k} \rangle \oplus \mathsf{sgn}\left[X[\mathbf{k}]\right] \oplus Z_p, \quad p \in [P], \tag{25}$$

where Z_p *is a Bernoulli random variable with probability upper bounded as* $\mathbb{P}_{\mathrm{e}} = e^{-\frac{\eta}{2}\mathrm{SNR}}$.

Proof. See Appendix C. □

From Proposition 3, it can be seen that the sign vector of the bin observation vector $\boldsymbol{U}$ can be viewed as some potentially corrupted bits received over a binary symmetric channel (BSC). The design of the offset matrix $\mathbf{D}$ for reliable and fast decoding over the BSC is thus the key to achieving sub-linear complexity.

In the following, we first present the sub-linear time *NSO-SPRIGHT Algorithm* that is easy to implement (i.e. a majority vote) and achieves a sub-linear complexity $T = O(K \log^3 N)$ with a sample cost of $M = O(K \log^2 N)$. Then, we present the sub-linear time *SO-SPRIGHT Algorithm* that maintains the optimal sample cost $M = O(K \log N)$ and simultaneously achieves sub-linear complexity $T = O(K \log N)$ using an iterative channel decoder.

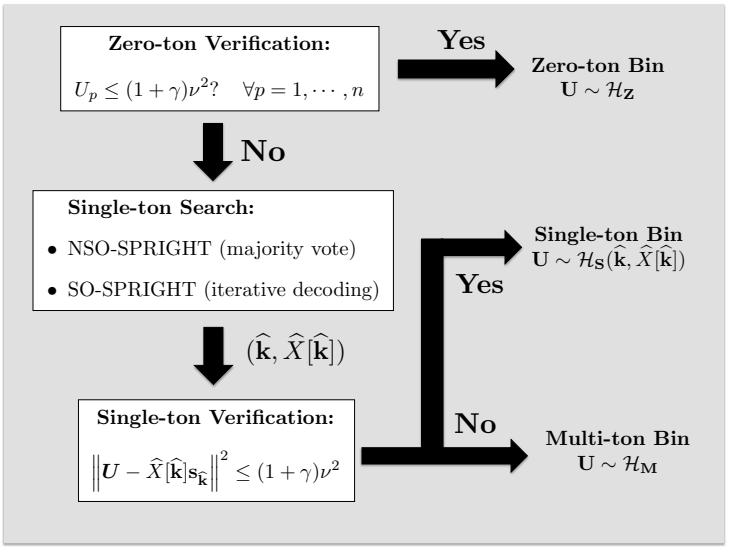


Figure 6: A simplified flowchart of the bin detection routine ψ for the noisy setting by choosing offsets according to Theorem 3 for the NSO-SPRIGHT and the SO-SPRIGHT algorithm.

5.3.1 The NSO-SPRIGHT Algorithm

Recall that the near-linear time design requires an exhaustive search due to the lack of structure of fully random offsets, which creates a bottleneck of the complexity. The key is to design a set of offsets that constitute a sufficiently good codebook to allow reliable transmissions of the n-bit index $\mathbf{k}$ over a BSC. In order to enable the bit-by-bit recovery of the binary representation of $\mathbf{k}$ as in the noiseless design, the first coding strategy we exploit is *repetition coding*, which is done by imposing structures on the random offsets for subsampling.

Definition 3. *Let $P = P_1 P_2$ with $P_1 = O(n)$ and $P_2 = n$. The NSO-SPRIGHT algorithm requires P_1 random offsets $\{\mathbf{d}_p\}_{p\in[P_1]}$ chosen independently and uniformly over $\mathbb{F}_2^n$ and P_2 modulated offsets $\{\mathbf{d}_{p,q}\}_{q\in[P_2]}$ such that*

$$\mathbf{d}_{p,q} \oplus \mathbf{d}_p = \mathbf{e}_q, \quad q \in [P_2] \tag{26}$$

where $\mathbf{e}_q$ is the q-th column of the identity matrix.

Given the offsets chosen as Definition 3, we can identify the q-th bit of $\mathbf{k}$ by jointly considering P_1 observations associated with offsets $\mathbf{d}_{p,q}$ across $p \in [P_1]$. More specifically,

$$\mathsf{sgn}\left[U_{p,q}\right] = \langle \mathbf{d}_{p,q}, \mathbf{k} \rangle \oplus \mathsf{sgn}\left[X[\mathbf{k}]\right] \oplus Z_{p,q} \tag{27}$$

$$\mathsf{sgn}\left[U_p\right] = \langle \mathbf{d}_p, \mathbf{k} \rangle \oplus \mathsf{sgn}\left[X[\mathbf{k}]\right] \oplus Z_p. \tag{28}$$

Since $\mathbf{d}_{p,q} \oplus \mathbf{d}_p = \mathbf{e}_q$, we have P_1 corrupted versions of $k[q]$:

$$\mathsf{sgn}\left[U_{p,q}\right] \oplus \mathsf{sgn}\left[U_p\right] = \langle \mathbf{e}_q, \mathbf{k} \rangle \oplus Z'_{p,q} = k[q] \oplus Z'_{p,q}, \tag{29}$$

where $Z'_{p,q} = Z_p \oplus Z_{p,q}$ is another Bernoulli variable with $\theta = \Pr\left(Z'_{p,q} = 1\right) = 2\mathbb{P}_\mathrm{e}(1 - \mathbb{P}_\mathrm{e}) < 1/2$. Then the MLE of $k[q]$ given observations $\{\mathsf{sgn}\left[U_{p,q}\right] \oplus \mathsf{sgn}\left[U_p\right]\}_{p=1}^{P_1}$ can be obtained as

$$\widehat{k}[q] = \arg\max_a \prod_{p=1}^{P_1} \theta^{\mathsf{sgn}[U_{p,q}] \oplus \mathsf{sgn}[U_p] \oplus a} (1-\theta)^{1 - \mathsf{sgn}[U_{p,q}] \oplus \mathsf{sgn}[U_p] \oplus a}. \tag{30}$$

Using the fact that $\theta < 1/2$ such that $\log(\theta/1-\theta) < 0$, we can simplify the objective as

$$\widehat{k}[q] = \arg\min_{a \in \mathbb{F}_2} \sum_{p=1}^{P_1} \mathsf{sgn}\left[U_{p,q}\right] \oplus \mathsf{sgn}\left[U_p\right] \oplus a. \tag{31}$$

In other words, the decoding scheme for the q-th bit of the index $\mathbf{k}$ becomes a simple *majority test* by accumulating $P_1 = O(n)$ random signs $\mathsf{sgn}\left[U_{p,q}\right] \oplus \mathsf{sgn}\left[U_p\right]$. Using the estimated bits $\{\widehat{k}[q]\}_{q\in[P_2]}$ together with $\mathbf{M}_c^T\mathbf{k} = \boldsymbol{j}$, the estimate $\widehat{\mathbf{k}}$ can be obtained accordingly. Finally, the value of the coefficient is obtained as (22). The zero-ton and single-ton verifications can be performed directly using the measurements associated with offsets $\mathbf{d}_p$ since there are $P_1 = O(n) = O(\log N)$ such random offsets, which have been shown to achieve high probability of success in the near-linear time design.

From Definition 3, we can see that there are a total of $P_1P_2 = O(n^2)$ offsets, and therefore each bin has $O(\log^2 N)$ observations. As a result, the NSO-SPRIGHT algorithm leads to a sample cost of $M = C\eta K \log^2 N = O(K\log^2 N)$. In terms of complexity, the majority vote requires $O(\log^2 N)$ operations for each bin, contributing to a total of $O(K\log^2 N)$ operations across all $O(K)$ bins. However, this complexity is dominated by generating $P = P_1P_2 = \log^2 N$ basic observation sets from B-point WHTs, each imposing an extra complexity of $O(K\log K) = O(K\log N)$ because of $K = O(N^\delta)$. As a result, this gives a total complexity of $T = O(K\log^3 N)$.

5.3.2 The SO-SPRIGHT Algorithm

While the NSO-SPRIGHT algorithm exploits repetition codes induced by the random offsets to robustify the noisy performance, we can further use better error correction codes to guide the choice of offsets. This is slightly more difficult to implement in practice since the decoding requires channel decoder instead of a simple majority vote, but the resulting sample complexity and computational complexity are order-optimal.

Definition 4. *Let $P = \sum_{i=1}^3 P_i$ with $P_i = O(n)$ for $i = 1, 2, 3$. The SO-SPRIGHT algorithm requires P_1 random offsets $\mathbf{d}_p$ for $p = 1, \cdots, P_1$ chosen independently and uniformly at random over $\mathbb{F}_2^n$, and P_2 zero offsets $\mathbf{d}_p = \mathbf{0}$ for $p = P_1 + 1, \cdots, P_1 + P_2$, and finally P_3 coded offsets $\mathbf{d}_p$ for $p = P_1 + P_2 + 1, \cdots, P$ such that the offset matrix $\mathbf{G} = [\cdots; \mathbf{d}_p; \cdots;] \in \mathbb{F}_2^{P_3\times n}$ constitutes a generator matrix of some linear block code with a minimum distance βP_3 with $\beta > \mathbb{P}_{\mathrm{e}}$.*

Recall Proposition 3, the observations associated with the coded offsets $\mathbf{G}$ can be written as

$$\begin{bmatrix} \mathsf{sgn}\left[U_{P_1+P_2+1}\right] \\ \vdots \\ \mathsf{sgn}\left[U_P\right] \end{bmatrix} = \mathbf{G}\mathbf{k} \oplus \mathsf{sgn}\left[X[\mathbf{k}]\right] \oplus \begin{bmatrix} Z_{P_1+P_2+1} \\ \vdots \\ Z_P \end{bmatrix}. \tag{32}$$

Note that there is a nuisance sign $\mathsf{sgn}\left[X[\mathbf{k}]\right]$ which is unknown to the robust bin detector. To illustrate our scheme, we first assume that there is a genie that informs the decoder of the sign of the coefficient $\mathsf{sgn}\left[X[\mathbf{k}]\right]$, and then we discuss how to get rid of the genie.

- *when* $\mathsf{sgn}\left[X[\mathbf{k}]\right]$ *is known a priori*: in this case, we can easily obtain

$$\begin{bmatrix} \mathsf{sgn}\left[U_{P_1+P_2+1}\right] \oplus \mathsf{sgn}\left[X[\mathbf{k}]\right] \\ \vdots \\ \mathsf{sgn}\left[U_P\right] \oplus \mathsf{sgn}\left[X[\mathbf{k}]\right] \end{bmatrix} = \mathbf{G}\mathbf{k} \oplus \begin{bmatrix} Z_{P_1+P_2+1} \\ \vdots \\ Z_P \end{bmatrix}. \tag{33}$$

  Since there are n information bits in the index $\mathbf{k}$, then there exists some channel code (i.e. $\mathbf{G}$) with block length $P_3 = n/R(\beta)$ that achieves a minimum distance of βP_3, where $R(\beta)$ is the rate of the code. As long as $\beta > \mathbb{P}_{\mathrm{e}}$, it is obvious that the unknown $\mathbf{k}$ can be decoded with exponentially decaying probability of error. There exist many codes that satisfy the minimum distance properties, but the concern is the decoding time. It is desirable to have decoding time linear in the block length so that the sample complexity and computational complexity can be maintained at $O(n)$, same as the noiseless case. Excellent examples include the class of expander codes or LDPC codes that allow for linear time decoding.

- *when* $\mathsf{sgn}\left[X[\mathbf{k}]\right]$ *is not known a priori*: we consider the observations associated with all the zero offsets $\mathbf{d}_p = \mathbf{0}$ for $p = P_1 + 1, \cdots, P_1 + P_2$

$$\begin{bmatrix} \mathsf{sgn}\left[U_{P_1+1}\right] \\ \vdots \\ \mathsf{sgn}\left[U_{P_1+P_2}\right] \end{bmatrix} = \mathsf{sgn}\left[X[\mathbf{k}]\right] \oplus \begin{bmatrix} Z_{P_1+1} \\ \vdots \\ Z_{P_1+P_2} \end{bmatrix} \tag{34}$$

  which can recover the sign correctly $\widehat{\mathsf{sgn}}\left[X[\mathbf{k}]\right] = \mathsf{sgn}\left[X[\mathbf{k}]\right]$ with high probability using a majority test (assuming $\mathbb{P}_{\mathrm{e}} \leq 1/2$). If $\mathbb{P}_{\mathrm{e}} > 1/2$, the sign is obtained accordingly using a minority test. Then we can proceed as if the sign is known a priori:

$$\begin{bmatrix} \mathsf{sgn}\left[U_{P_1+P_2+1}\right] \oplus \widehat{\mathsf{sgn}}\left[X[\mathbf{k}]\right] \\ \vdots \\ \mathsf{sgn}\left[U_P\right] \oplus \widehat{\mathsf{sgn}}\left[X[\mathbf{k}]\right] \end{bmatrix} = \mathbf{G}\mathbf{k} \oplus \begin{bmatrix} Z_{P_1+P_2+1} \\ \vdots \\ Z_P \end{bmatrix}. \tag{35}$$

Finally, the value of the coefficient is obtained as (22). The zero-ton and single-ton verifications can be performed directly using the observations associated with offsets $\mathbf{d}_p$. Since there are $P_1 = O(n) = O(\log N)$ such random offsets, which have been shown to achieve high probability of success in the near-linear time design.

Using the SO-SPRIGHT design, we can see that there are three sets of offsets, where one set includes $P_3 = O(n)$ offsets for the single-ton search, and the second set includes $P_2 = O(n)$ zero offsets for the sign reference, and $P_1 = O(n)$ random offsets for the zero-ton and single-ton verifications. Therefore, we have a total of $P = \sum_{i=1}^{3} P_i = O(n) = O(\log N)$ offsets and each bin has $O(\log N)$ observations. As a result, the SO-SPRIGHT algorithm leads to a sample cost of $M = C\eta KP = O(K \log N)$, which is the same as the noiseless case [2]. In terms of complexity, if $\mathbf{G}$ is a properly chosen channel code generator matrix from the class of expander codes or LPDC codes, the decoding time for the index requires $O(n) = O(\log N)$ operations for each bin. This contributes to a total of $O(K \log N)$ complexity across all $O(K)$ bins. However, this complexity is dominated by subsampling for generating P basic observation sets from B-point WHTs, each imposing an extra complexity of $O(K \log K) = O(K \log N)$ because of $K = O(N^{\delta})$. As a result, this gives a total complexity of $T = O(K \log^2 N)$, which is also the same as the noiseless case [2].

6 Applications

In the following, we provide some machine learning concepts that can be cast as a WHT computation or expansion.

Example 2 (Pseudo-Boolean Function and Sparse Polynomial)**.** *An arbitrary pseudo-Boolean function can be represented uniquely by a multi-linear polynomial over the hypercube* $(z_1, \cdots, z_n) \in \{-1, +1\}^n$*:*

$$f(z_1, \cdots, z_n) = \sum_{\mathcal{S} \subseteq [n]} \alpha_{\mathcal{S}} \prod_{i \in \mathcal{S}} z_i, \ \forall\, z_i \in \{-1, +1\}, \tag{36}$$

where $\mathcal{S}$ *is a subset of* $[n] := \{1, \cdots, n\}$*, and* $\alpha_{\mathcal{S}}$ *is the Walsh (Fourier) coefficient associated with the monomial* $\prod_{i \in \mathcal{S}} z_i$*. If we replace* z_i *by* $(-1)^{m[i]}$ *such that* $z_i = -1$ *when* $m[i] = 1$ *and* $z_i = 1$ *when* $m[i] = 0$*, we have* $x[\mathbf{m}] = f\left((-1)^{m[1]}, \cdots, (-1)^{m[n]}\right)$ *for* $\mathbf{m} \in \mathbb{F}_2^n$ *and* $X[\mathbf{k}] = \sqrt{N}\alpha_{\mathcal{S}}$ *such that* $\mathrm{supp}\,(\mathbf{k}) = \mathcal{S}$*.*

Example 3 (Set Functions)**.** *A set function is an arbitrary real-valued function* $f : 2^{[n]} \to \mathbb{R}$ *defined for every element in the power set* $\mathcal{Z} \in 2^{[n]}$*, which has a Walsh expansion given by*

$$f(\mathcal{Z}) = \frac{1}{\sqrt{N}} \sum_{\mathcal{S} \in 2^{[n]}} \hat{f}(\mathcal{S})(-1)^{|\mathcal{S} \cap \mathcal{Z}|}, \tag{37}$$

where $\hat{f}(\mathcal{S})$ is the Walsh (Fourier) coefficient. Clearly, a set function can also be viewed as a n-ary pseudo-Boolean function in (36) *such that $f(\mathcal{Z}) = f(z_1, \cdots, z_n)$ as long as $z_i = -1$ if $i \in \mathcal{Z}$ and $z_i = 1$ if $i \notin \mathcal{Z}$. Therefore, each function value $f(\mathcal{Z})$ can be regarded as a sample $x[\mathbf{m}] = f(\text{supp}\,(\mathbf{m}))$, where the Walsh coefficient satisfies $X[\mathbf{k}] = \hat{f}(\mathcal{S})$ as long as $\text{supp}\,(\mathbf{k}) = \mathcal{S}$.*

Example 4 (Decision Tree Learning)**.** *Decision trees are machine-learning methods for constructing prediction models from data, whose goal is to predict the value of a target label f based on n input variables $z_i \in \{\pm 1\}$ for $i \in [n]$. More specifically, this includes classification trees (discrete-valued outcome $f \in \mathbb{Z}$) and regression trees (real-valued outcome $f \in \mathbb{R}$). Decision tree models are usually constructed from top-down starting at the root node, by choosing a certain variable z_i for some i at each step that optimally splits the set of training data with respect to some measure of goodness. Hence, for each set of input variables $(z_1, \cdots, z_n) \in \{-1, +1\}^n$, there is a unique leaf node in the tree that assigns the target label f. This is mathematically equivalent to learning a (pseudo)-Boolean function, which can be cast as a problem of computing the WHT of f.*

It has been found that many instances of the examples above exhibit sparsity in the Walsh spectrum. In general, our SPRIGHT framework can be applied to learning K-sparse pseudo-Boolean polynomials $f : \{\pm 1\}^n \to \mathbb{R}$ with n variables. A concrete example is in decision tree learning, where the underlying (pseudo)-Boolean function has a sparse spectrum if the decision tree has few leaf nodes with short depth. An extreme case would be when the underlying function only depends on few input variables, which is also referred to as the juntas problem in Boolean analysis[5]. Therefore, if the K-sparse N-point WHT can be computed efficiently, these machine learning applications can benefit greatly from the reductions in both the sample complexity and computational complexity. In the following, we present a specific machine learning application in graph sketching.

6.1 Applications in Hypergraph Sketching

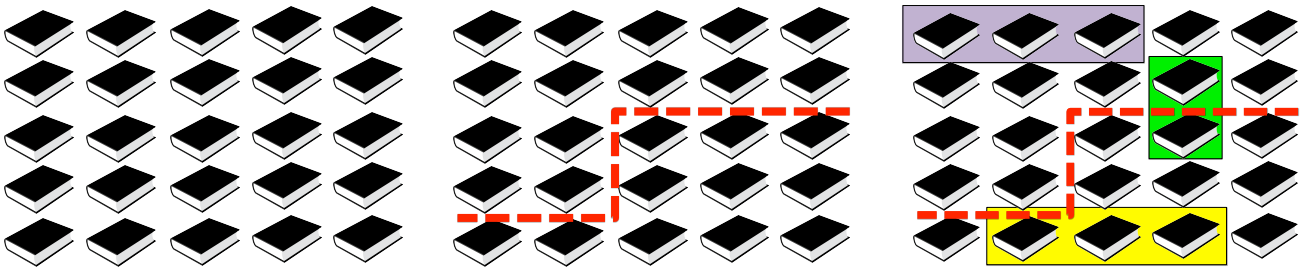
A *hypergraph*, denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is a generalized notion of graphs where each edge $e \in \mathcal{E}$, called the *hyperedges*, can connect more than two nodes in the node set $\mathcal{V}$. Hypergraph sketching here refers to the procedure of identifying the unknown hypergraph structure from cut queries. Hypergraphs have been very useful in relational learning, which has received extensive attentions in recent years since many real-world data are organized by the relations between entities. Some of the interesting problems involved in relational learning include the discovery of communities, classification, and predictions of possible new relations.

We describe the hypergraph sketching application through an example depicted in Fig. 7. Consider a scenario where there are n books from a certain provider (e.g. Amazon) and each book is characterized by a node in the graph. There are numerous transactions taking place in which each customer buys a few books. In this setting, the relationship between books in each transaction can be captured by a hyperedge, which connects the subset of books bought in the same transaction. A cartoon illustration is depicted in Fig. 7, where there are 3 distinct sets of books bought in different transactions with each set coded in different colors. Then, the hypergraph sketching problem is equivalent to solving the following problem under a the following *query* model:

- Pick an arbitrary partition $(S, \bar{S})$ of n books such that $S \cup \bar{S} = \mathcal{V}$ (see Fig. 7(b)).
- One can query the following: i) are there any transactions that include books from both sets $(S, \bar{S})$? and ii) if there are, what is the total number of transactions that satisfy this requirement? For example in Fig. 7(c), the resulting query would return 1 since there is only 1 transaction that includes books from both sets.
- How many such queries are needed to fully learn all the unknown distinct subsets of books that are bought in different transactions?

Note that the query requested here is in fact the number of hyperedges that cross over the two sets $(S, \bar{S})$, which is defined as the *cut value* of the graph. As shown next, this can be mathematically established as a sparse WHT computation problem, where our SPRIGHT framework is found to be useful.

[5]It is well-known that learning juntas using *random* samples is NP-hard. Our framework tackles the juntas problem using *specifically chosen* samples, and hence we can achieve sub-linear sample cost and run-time. This is not a contradiction.



(a) Hidden graph of n books: there are a few purchase patterns, where each corresponds to a hyperedge

(b) Pick some partition $(S, \bar{S})$: how many transactions include books from both sets $(S, \bar{S})$?

(c) Query: in this example, the query result for this partition is 1 and the graph has 3 distinct subsets.

Figure 7: Given a set of n books, infer the graph structure by querying graph cuts.

Let $|\mathcal{V}| = n$ and $|\mathcal{E}| = s$. A cut $\mathcal{S} \subseteq \mathcal{V}$ is a set of selected vertices, denoted by the binary n-tuple $\mathbf{m} = [m[1], \cdots, m[n]]$ over $\mathbb{F}_2^n$, where $m[i] = 1$ if $i \in \mathcal{S}$ and $m[i] = 0$ if $i \notin \mathcal{S}$. The cut value $x[\mathbf{m}]$ for a specific cut $\mathbf{m}$ in the hypergraph is defined as $x[\mathbf{m}] = \left|\{e \in \mathcal{E} : e \cap \mathcal{S} \neq \varnothing,\ e \cap \bar{S} \neq \varnothing\}\right|$, where $\bar{S} = \mathcal{V}/\mathcal{S}$. In other words, the cut value corresponds to the number of hyperedges that crosses between the two sets $(S, \bar{S})$. Given a partition $\mathbf{m} \in \mathbb{F}_2^n$, for some edge $e \in \mathcal{E}$, we define the following function to indicate whether it crosses over two sets $(S, \bar{S})$:

$$1_e[\mathbf{m}] = \prod_{i \in e} \frac{\left(1 + (-1)^{m[i]}\right)}{2} + \prod_{i \in e} \frac{\left(1 - (-1)^{m[i]}\right)}{2}. \tag{38}$$

For example, if all the nodes connected through this particular hyperedge $i \in e$ is on the same side of the partition $(S, \bar{S})$, which implies that either $m[i] = 0$ or $m[i] = 1$ for all $i \in e$, this indicator $1_e[\mathbf{m}] = 1$ is 1. This suggests that when the edge e does *not* cross over the two sets $(S, \bar{S})$, the indicator takes the value 1. Therefore, the total count of edges that do cross over can be obtained accordingly as

$$x[\mathbf{m}] = \sum_{e \in \mathcal{E}} \left(1 - 1_e[\mathbf{m}]\right). \tag{39}$$

By substituting $1_e[\mathbf{m}]$ with (38), it can be equivalently written as a WHT expansion as follows:

$$x[\mathbf{m}] = \sum_{\mathbf{k} \in \mathbb{F}_2^n} X[\mathbf{k}](-1)^{\langle \mathbf{k}, \mathbf{m} \rangle}, \tag{40}$$

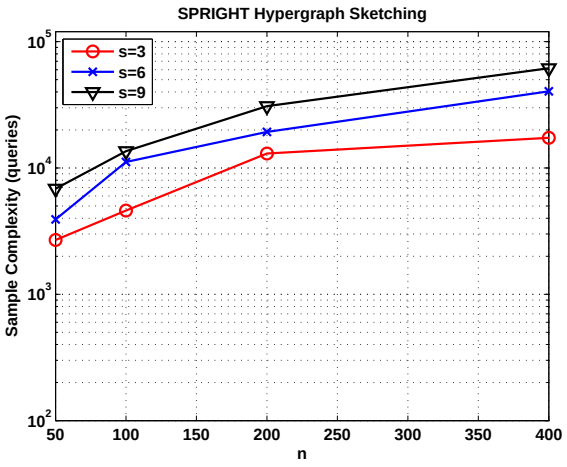
where the coefficient $X[\mathbf{k}]$ is a scaled WHT coefficient such that $X[\mathbf{0}] = \left(s - \sum_{e \in \mathcal{E}} \frac{1}{2^{|e|-1}}\right)$ and

$$X[\mathbf{k}] = \begin{cases} \frac{1}{2^{|e|-1}}, & \text{if } \operatorname{supp}(\mathbf{k}) \in e \text{ and } |\operatorname{supp}(\mathbf{k})| \text{ is even} \\ 0, & \text{otherwise} \end{cases} \tag{41}$$

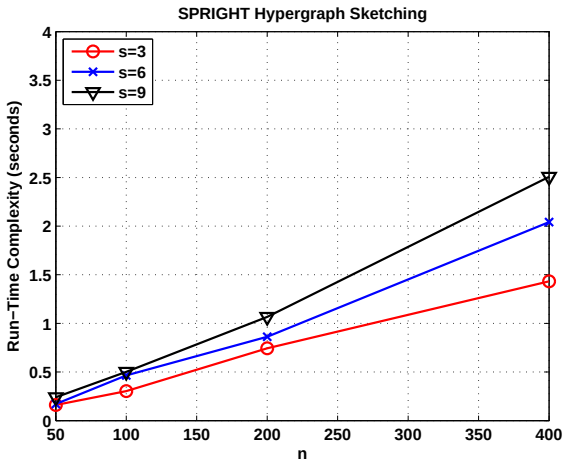
Clearly, if the number of hyperedges is small $s \ll 2^n$ and the maximum size of each hyperedge is small, the coefficients $X[\mathbf{k}]$'s are sparse. For example, if the hyperedge size can be universally bounded by d, the sparsity can be well upper bounded by $K \leq s2^{d-1}$.

6.2 Simple Experiment

Here we consider the noiseless scenario as a proof of concept, we use our SO-SPRIGHT algorithm for hypergraph sketching, which requires $O(Kn)$ queries for interpolating the total 2^n cut values with run-time $O(Kn^2)$. In this experiment, we randomly generate hypergraphs with $n = 50$ to 400 nodes with $s = 3, 6, 9$ edges, where each edge does not connect more than $d = 6$ nodes. As can be seen, our SPRIGHT framework computes the sparse coefficients $X[\mathbf{k}]$ in time $\Theta(K \log Kn) = \Theta(Kn^2)$ from only $\Theta(Kn)$ cut queries.



(a) Query cost scaling with the graph size n



(b) Run-time scaling with the graph size n

7 Numerical Experiments

In this section, we test the NSO-SPRIGHT algorithm and SO-SPRIGHT algorithm respectively. We first showcase the performances in many settings by varying the signal length $N = 2^n$, sparsity and SNR. Then, we demonstrate possible applications of our SPRIGHT framework in machine learning domains such as hypergraph sketching and decision tree learning over large datasets.

7.1 Performance of the SPRIGHT Framework

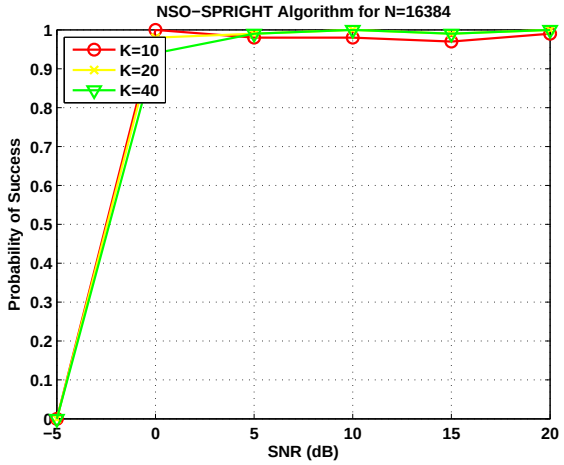
Here, we synthetically generate time domains samples $\mathbf{x}$ from a K-sparse WHT signal $\mathbf{X}$ of length $N = 2^n$ with K randomly positioned non-zero coefficients of magnitude $\pm\rho$. The setup of our experiments is given below:

- *subsampling parameters*: we fix the number of groups to $C = 3$ and the number of bins in each group is $B = 2^b$ where $b = \lceil \log_2(K) \rceil$. Note that in this case $B \approx K$ and thus $\eta \approx 1$.
- *NSO-SPRIGHT algorithm parameters*: we choose $P_1 = 2n$ random offsets and $P_2 = n$ modulated offsets. Thus the sample cost is $M_{\text{NSO}} = 2CBn^2 \approx 6Kn^2$ and the complexity is $T_{\text{NSO}} = O(Kn^3)$.
- *SO-SPRIGHT algorithm parameters*: we choose $P_1 = 2n$ coded offsets for the single-ton search, $P_2 = n$ zero offsets and $P_3 = n$ random offsets for the zero-ton and single-ton verifications. For the single-ton search, the $P_1 = 2n$ coded offsets are chosen to induce a $(3, 6)$-regular LDPC code, where the search utilizes the Gallager's bit flipping algorithm for decoding, which imposes linear run-time $O(n)$. The sample cost is $M_{\text{SO}} = 4CBn \approx 12Kn$ and the complexity is $T_{\text{SO}} = O(Kn^2)$.

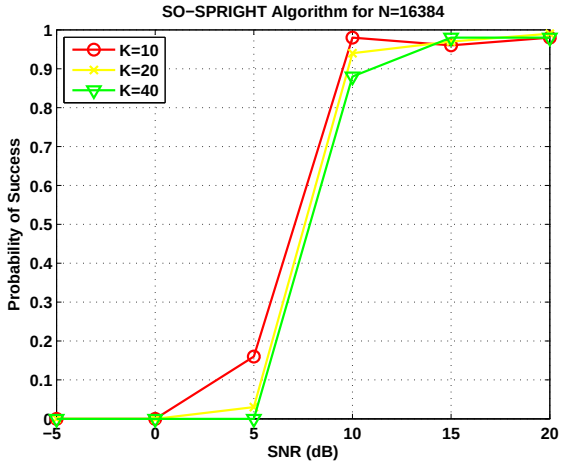
7.1.1 Noise Robustness

In this subsection, we compare the noise robustness of the NSO-SPRIGHT and SO-SPRIGHT algorithms. The experiment settings are given below:

- *input profile*: we generate a sparse WHT vector or length $N = 2^n$ with $n = 14$ and $K = 10, 20, 40$ non-zero coefficients respectively. Therefore, the signal dimension is $N = 16384$. The non-zero WHT coefficients are chosen with uniformly random support and random amplitudes $\{\pm 1\}$. The input signal samples $\mathbf{x}$ is obtained by taking the inverse WHT of the sparse WHT vector and adding i.i.d. Gaussian noise samples with variance σ^2 determined by the range of $\mathsf{SNR} = [-5 : 5 : 20]$ dB .



(c) Probability of success versus SNR



(d) Probability of success versus SNR

Note that the sample complexity of the NSO-SPRIGHT algorithm is approximately a factor of n more than the SO-SPRIGHT algorithm, and thus the recovery performance is better under the same experiment setup. However, this is due to our simple choice of $(3, 6)$-regular LDPC codes for inducing the offsets in the SO-SPRIGHT algorithm, which is far from capacity-achieving. Potentially one can use better LDPC code ensembles or even spatially coupled LDPC codes to provide better performance at the low SNR regime. Here the $(3, 6)$-regular ensemble is simply an example to showcase the algorithm.

7.1.2 Sample Complexity and Run-time Performance

In this subsection, we compare the sample complexity and run-time performance of the NSO-SPRIGHT and SO-SPRIGHT algorithms. The experiment settings are given below:

- *input profile*: we generate a sparse WHT vector or length $N = 2^n$ with $K = 10, 20, 40$ non-zero coefficients respectively and vary n from $n = 7$ to $n = 17$. Therefore, the signal dimension spans from $N = 128 \approx 10^2$ to $131072 \approx 0.1 \times 10^6$. The non-zero WHT coefficients are chosen with uniformly random support and random amplitudes $\{\pm 1\}$. The input signal samples $\mathbf{x}$ is obtained by taking the inverse WHT of the sparse WHT vector and adding i.i.d. Gaussian noise samples with variance σ^2 determined by the $\mathsf{SNR} = 10$ dB.
- *benchmark*: as the signal length $N = 2^n$ varies, the algorithm parameters are fixed over 200 random experiments. We record a data point only when the success probability exceeds 0.95.

8 Conclusions

In this paper, we have proposed the SPRIGHT framework to compute a K-sparse N-point WHT, where the NSO-SPRIGHT algorithm uses $O(K \log^2 N)$ samples and $O(K \log^3 N)$ operations while the SO-SPRIGHT algorithm maintains the optimal sample scaling $O(K \log N)$ and complexity $O(K \log^2 N)$ as that of the noiseless case. Our approach is based on strategic subsampling of the input noisy samples using a small set of randomly shifted patterns that are carefully designed, which achieves a vanishing failure probability.

References

[1] S. Pawar and K. Ramchandran, "Computing a k-sparse n-length discrete Fourier transform using at most $4k$ samples and $\mathcal{O}(k \log k)$ complexity," *arXiv preprint arXiv:1305.0870*, 2013.

[2] R. Scheibler, S. Haghighatshoar, and M. Vetterli, "A fast hadamard transform for signals with sub-linear sparsity," *arXiv preprint arXiv:1310.1803*, 2013.

[3] W. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *Proceedings of the IEEE*, vol. 57, no. 1, pp. 58–68, 1969.

[4] T. R. WGI, "Spreading and modulation (fdd)," 3GPP Tech Rep. TS25. 213, 2000. http://www. 34gpp. org, Tech. Rep.

[5] S. Haghighatshoar and E. Abbe, "Polarization of the rényi information dimension for single and multi terminal analog compression," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 779–783.

[6] M. Lee and M. Kaveh, "Fast hadamard transform based on a simple matrix factorization," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, no. 6, pp. 1666–1667, Dec 1986.

[7] J. Johnson and M. Puschel, "In search of the optimal walsh-hadamard transform," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, vol. 6, 2000, pp. 3347–3350 vol.6.

[8] K. J. Horadam, *Hadamard matrices and their applications*. Princeton university press, 2007.

[9] A. Hedayat and W. Wallis, "Hadamard matrices and their applications," *The Annals of Statistics*, vol. 6, no. 6, pp. 1184–1238, 1978.

[10] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Simple and practical algorithm for sparse fourier transform," in *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2012, pp. 1183–1194.

[11] ——, "Nearly optimal sparse fourier transform," in *Proceedings of the 44th symposium on Theory of Computing*. ACM, 2012, pp. 563–578.

[12] B. Ghazi, H. Hassanieh, P. Indyk, D. Katabi, E. Price, and L. Shi, "Sample-optimal average-case sparse fourier transform in two dimensions," *arXiv preprint arXiv:1303.1209*, 2013.

[13] M. Iwen, A. Gilbert, and M. Strauss, "Empirical evaluation of a sub-linear time sparse dft algorithm," *Communications in Mathematical Sciences*, vol. 5, no. 4, pp. 981–998, 2007.

[14] M. A. Iwen, "Combinatorial sublinear-time fourier algorithms," *Foundations of Computational Mathematics*, vol. 10, no. 3, pp. 303–338, 2010.

[15] S. A. Pawar, "Pulse: Peeling-based ultra-low complexity algorithms for sparse signal estimation," *PhD Dissertation*, 2013.

[16] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss, "Near-optimal sparse fourier representations via sampling," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM, 2002, pp. 152–161.

[17] A. C. Gilbert, S. Muthukrishnan, and M. Strauss, "Improved time bounds for near-optimal sparse fourier representations," in *Optics & Photonics 2005*. International Society for Optics and Photonics, 2005, pp. 59 141A–59 141A.

[18] Y. Mansour, "Randomized interpolation and approximation of sparse polynomials," *SIAM Journal on Computing*, vol. 24, no. 2, pp. 357–368, 1995.

[19] A. C. Gilbert, M. J. Strauss, and J. A. Tropp, "A tutorial on fast fourier sampling," *Signal processing magazine, IEEE*, vol. 25, no. 2, pp. 57–66, 2008.

[20] M. Cheraghchi and P. Indyk, "Nearly optimal deterministic algorithm for sparse walsh-hadamard transform," *arXiv preprint arXiv:1504.07648*, 2015.

[21] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 569–584, 2001.

[22] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 599–618, 2001.

[23] R. Pedarsani, K. Lee, and K. Ramchandran, "Phasecode: Fast and efficient compressive phase retrieval based on sparse-graph-codes," *arXiv preprint arXiv:1408.0034*, 2014.

[24] M. Bouvel, V. Grebinski, and G. Kucherov, "Combinatorial search on graphs motivated by bioinformatics applications: A brief survey," in *Graph-Theoretic Concepts in Computer Science*. Springer, 2005, pp. 16–27.

[25] L. Birgé, "An alternative point of view on lepski's method," *Lecture Notes-Monograph Series*, pp. 113–133, 2001.

Appendices

A Proof of Theorem 1

From Theorem 2, it is shown that as long as $C \leq 8$ groups and $B = O(K)$, the oracle-based peeling decoder succeeds with probability at least $1 - O(1/K)$ for $0 < \delta < 1$. In Theorem 3, it is further shown that with the proposed bin detection routine using P observation sets (chosen differently) in each group, the peeling decoder continues to succeed with probability at least $1 - O(1/K)$ in the presence of noise. Therefore, the sample complexity is $M = CBP = O(KP)$. On the other hand, the computational complexities stem from two sources:

- The computation of B-point WHTs for subsampling: there are P observations sets in each group, where each observation set requires a B-point WHT. Thus the total complexity is $O(PB \log B) = O(PK \log N)$, where $K = O(N^{\delta})$ has been used;
- The bin detection routine in each peeling iteration for decoding: In the NSO-SPRIGHT scheme it is a majority vote, which leads to a complexity of $O(P)$. In the SO-SPRIGHT scheme it requires the decoding of a linear code formed by the P offsets. As mentioned, one can potentially use (spatially coupled) LDPC or expander codes to achieve linear-time decoding $O(P)$, where P is the block length of the code. Therefore, both sub-linear detection schemes result in a total complexity of $O(KP)$ throughout the $O(K)$ peeling iterations.

Clearly, the complexity is dominated by the subsampling $T = O(PK \log N)$. Substituting the corresponding P required by the sub-linear bin detection routines in the NSO-SPRIGHT and the SO-SPRIGHT schemes, we arrive at our stated results.

B Proof of Theorem 2 : Oracle-based Peeling Decoder Analysis

B.1 Design and Analysis for the Very Sparse Regime $0 < \delta \leq 1/3$

To keep our discussions general, we choose C subsampling groups and $B = 2^b$ with $b = \delta n$ such that $B = \eta K$ for some $\eta > 0$ and the subsampling matrices

$$\mathbf{M}_c = [\mathbf{0}^T_{(c-1)\times b}, \mathbf{I}^T_{b\times b}, \mathbf{0}^T_{(n-cb)\times b}]^T, \quad c \in [C], \tag{42}$$

which freezes a $(n-b)$-bit segment of the time domain indices $\mathbf{m} \in \mathbb{F}_2^n$ to all zeros[6]. Then, each left node labeled $\mathbf{k} \in \mathbb{F}_2^n$ is connected to a right node labeled $\boldsymbol{j} \in \mathbb{F}_2^b$ determined by the aliasing pattern $\mathbf{M}_c^T\mathbf{k} = \boldsymbol{j}$. Therefore, the graph ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ in Definition 1 is consistent with the "*balls-and-bins*" model, where the $\mathbf{k}$-th ball (i.e. left node $\mathbf{k}$) is thrown to bin $\boldsymbol{j}_c = \mathcal{H}_c(\mathbf{k})$ in group c. Now we show that given the uniform support distribution, the graph ensemble is further consistent with the random "balls-and-bins" model in each group.

We divide the index $\mathbf{k}$ into $C+1$ segments as $\mathbf{k} = [\boldsymbol{k}_1^T, \boldsymbol{k}_2^T, \cdots, \boldsymbol{k}_{C-1}^T, \boldsymbol{k}_C^T, \boldsymbol{k}_{C+1}^T]^T$, where each of the first C segments $\boldsymbol{k}_c = [k[cb], \cdots, k[(c-1)b+1]]^T$ for $c \in [C]$ contains b bits while the last segment $\boldsymbol{k}_{C+1} = [k[n], \cdots, k[Cb+1]]^T$ contains the remaining $(n-Cb)$ bits. Then, the hash functions associated with the subsampling matrices in (42) are $\mathcal{H}_c(\mathbf{k}) = \mathbf{M}_c^T\mathbf{k} = \boldsymbol{k}_c$, which sifts out the b-bit segment $\boldsymbol{k}_c$ independently out of n bits from the index $\mathbf{k}$ in group c. We call the output of the hash function in each group the *bit segmentation*. Clearly, these *bit segmentations* can be chosen differently according to the choice of subsampling matrices $\{\mathbf{M}_c\}_{c\in[C]}$. For example, the *bit segmentations* in the first 3 groups are

$$\boldsymbol{j}_1 = \begin{bmatrix} k[1] \\ \vdots \\ k[b] \end{bmatrix}, \quad \boldsymbol{j}_2 = \begin{bmatrix} k[b+1] \\ \vdots \\ k[2b] \end{bmatrix}, \quad \boldsymbol{j}_3 = \begin{bmatrix} k[2b+1] \\ \vdots \\ k[3b] \end{bmatrix}. \tag{43}$$

[6]The reason for $\delta = 1/3$ to be the separation point between the very sparse regime and the less sparse regime will become clear in Proposition 4 in the following section, where $C \geq 3$ is proven necessary for successful decoding with high probability. With the requirement $C \geq 3$ and the constraint $Cb \leq n$ due to the choice of $\mathbf{M}_c$, we have $b = \delta n$ and therefore $\delta \leq 1/3$.

Since each element $\mathbf{k}$ of the support set $\mathcal{K}$ is chosen independently and uniformly at random from $\mathbb{F}_2^n$ by Assumption 1, each *bit segmentation* $\boldsymbol{j}_c = \mathcal{H}_c(\mathbf{k})$ is independently and uniformly chosen from $\{0,1\}$ for each ball. Therefore, each left ball is thrown independently into the bins on the right, which suggests that the edges from each left node to each right node are connected independently. Further, the bin index in each group $\boldsymbol{j}_c$ contains bit segments in $\mathbf{k}$ that are uniformly distributed, and hence each ball is thrown uniformly at random to one of the B right nodes in that group.

In the following, we show that if the redundancy parameter $\eta = B/K$ is chosen appropriately for the graph ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ with C subsampling groups and $\mathbf{M}_c$ chosen as (42), then given the oracle, all the edges of the graph can be peeled off in $O(K)$ peeling iterations with high probability.

Proposition 4 (**Oracle-based Peeling Decoder Performance for** $0 < \delta \leq 1/3$)**.** *If we use $C = 3$ groups with the set size $B = 0.4073K$, where the subsampling matrices $\mathbf{M}_c$ for each group are chosen as in* (42)*, the induced graph ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ guarantees that the oracle-based peeling decoder peels off all the edges in $O(K)$ iterations with probability at least $1 - O(1/K)$.*

Proof. The proof is given in the following subsections. □

B.1.1 Density Evolution

Density evolution, a powerful tool in modern coding theory, tracks the average density of remaining edges that are not decoded after a fixed number of peeling iteration $i > 0$. We introduce the concept of *directed neighborhood* of a certain edge in the bipartite graph up to depth $\ell = 2i$. This concept is important in the density evolution analysis since the peeling of an edge in the i-th iteration depends solely on the removal of the edges from this neighborhood in the previous $i-1$ iterations. The *directed neighborhood* $\mathcal{N}_{\mathrm{e}}^{\ell}$ at depth ℓ of a certain edge $e = (v, c)$ is defined as the induced sub-graph containing all the edges and nodes on paths $e_1, \cdots, e_\ell$ starting at a variable node v (left node) such that $e_1 \neq e$. An example of a directed neighborhood of depth $\ell = 2$ is given in Fig. 9.

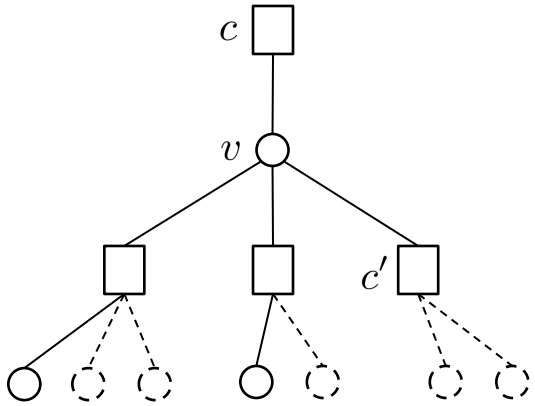


Figure 9: Directed neighborhood of depth 2 of an edge $\vec{e} = (v, c)$. The dashed lines correspond to nodes/edges removed at the end of iteration i. The edge between v and c can be potentially removed at iteration $i+1$ as one of the check nodes c' is a singleton (it has no more variable nodes remaining at the end of iteration i).

To analyze the performance of the peeling decoder over the bipartite graph, we need to understand the edge degree distributions on the left and right of the bipartite graph. Since the left edge degree distribution is already known due to the regularity of the graph ensemble induced by subsampling, next we study the right edge degree distribution.

Lemma 1. *Let ρ_j be the fraction of edges in the bipartite graph connecting to right nodes with degree j. In the very sparse regime $0 < \delta \leq 1/3$, if we use C subsampling groups with subsampling matrices $\{\mathbf{M}_c\}_{c\in[C]}$ chosen as* (42)*, the edge degree sequence ρ_j of the graph ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ is obtained as*

$$\rho_j = \frac{(1/\eta)^{j-1} e^{-1/\eta}}{(j-1)!}. \tag{44}$$

Proof. See Appendix B.4. □

Now let us consider the local neighborhood $\mathcal{N}_{\mathrm{e}}^{2i}$ of an arbitrary edge $e = (v, c)$ with a left regular degree d and right degree distribution given by $\{\rho_j\}_{j=1}^{K}$. If the sub-graph corresponding to the neighborhood $\mathcal{N}_{\mathrm{e}}^{2i}$ of the edge

$e = (v, c)$ is a *tree* or namely *cycle-free*, then the peeling procedures over different bins in the first i iterations are independent, which can greatly simplify our analysis. Density evolution analysis is based on the assumption that this neighborhood is cycle-free (tree-like), and we will prove later (in the next subsection) that all graphs in the regular ensemble behave like a tree when N and K are large and hence the actual density evolution concentrates well around the density evolution result.

Let p_i be the probability of this edge being present in the bipartite graph after $i > 0$ peeling iterations. If the neighborhood is a tree as in Fig. 9, the probability p_i can be written with respect to the probability p_{i-1} at the previous depth in a recursive manner $p_i = \left(1 - \sum_j \rho_j (1 - p_{i-1})^{j-1}\right)^{C-1}$ for $i = 1, 2, 3, \cdots$. The term $\sum_j \rho_j (1 - p_{i-1})^{j-1}$ can be approximated using the right degree generating polynomial

$$\rho(x) := \sum_j \rho_j x^{j-1} = e^{-(1-x)\frac{1}{\eta}}, \tag{45}$$

where we have used (44) to derive the second expression. Therefore, the density evolution equation for our peeling decoder can be obtained as

$$p_i = \left(1 - e^{-\frac{1}{\eta} p_{i-1}}\right)^{C-1}, i = 1, 2, 3, \cdots \tag{46}$$

Clearly, the probability p_i can be made arbitrarily small for a sufficiently large but finite $i > 0$ as long as C and η are chosen properly. One can find the minimum value η for a given C to guarantee $p_i < p_{i-1}$, which is shown in Table 1. Due to lack of space we only show up to $C = 6$.

C	2	3	4	5	6
η	1.0000	0.4073	0.3237	0.2850	0.2616
$C\eta$	2.0000	1.2219	1.2948	1.4250	1.5696

Table 1: Minimum value for η given the number of groups C

Lemma 2 (Density evolution $0 < \delta \leq 1/3$). *Let $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ be the graph ensemble induced by subsampling with C subsampling groups using subsampling matrices $\{\mathbf{M}_c\}_{c\in[C]}$ in* (42) *in the very sparse regime $0 < \delta \leq 1/3$, where the number of groups C and the redundancy parameter η chosen from Table 1. Denote by $\mathcal{T}_i$ the event where the local $2i$-neighorhood $\mathcal{N}_e^{2i}$ of every edge in the graph is tree-like and let Z_i be the total number of edges that are not decoded after i (an arbitrarily large but fixed) peeling iterations. For any $\varepsilon > 0$, there exists a finite number of iteration $i > 0$ such that*

$$\mathbb{E}[Z_i | \mathcal{T}_i] = KC\varepsilon/4, \tag{47}$$

where the expectation is taken with respect to the random graph ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$.

Proof. See Appendix B.5. □

Based on this lemma, we can see that if the bipartite graph has a local neighborhood that is tree-like up to depth $2i$ for every edge, the peeling decoder on average peels off all but an arbitrarily small fraction of the edges.

B.1.2 Convergence to Density Evolution

Given the mean performance analysis (in terms of the number of undecoded edges) over cycle-free graphs, now we provide a *concentration analysis* on the number of the undecoded edges Z_i for *any graph from the ensemble* $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ at the i-th iteration, by showing that Z_i converges to the density evolution.

Lemma 3 (**Convergence to density evolution for** $0 < \delta \leq 1/3$)**.** *Over the probability space of all graphs from* $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$*, let* p_i *be as given in the density evolution* (46)*. Given any* $\varepsilon > 0$ *and a sufficiently large* K*, there exists a constant* $c > 0$ *such that*

$$(\texttt{i}) \qquad \mathbb{E}[Z_i] < KC\varepsilon/2 \tag{48}$$

$$(\texttt{ii}) \qquad \Pr\left(|Z_i - \mathbb{E}[Z_i]| > KC\varepsilon/2\right) \leq 2\exp\left(-c\varepsilon^2 K^{\frac{1}{4i+1}}\right). \tag{49}$$

Proof. We provide a *concentration analysis* in Appendix B.6 on the number of the remaining edges for an *arbitrary graph from the ensemble* by showing that Z_i converges to the mean analysis result. Here is a sketch of the proof:

- *Mean analysis on general graphs from ensembles*: first, we use a counting argument similar to [23] to show that any random graph from the ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ behaves like a *tree* with high probability. Therefore, the expected number of remaining edges can be made arbitrarily close to the mean analysis $|\mathbb{E}[Z_i] - \mathbb{E}[Z_i|\mathcal{T}_i]| < KC\varepsilon/4$ such that $\mathbb{E}[Z_i] < KC\varepsilon/2$ if N and K are greater than some constants.
- *Concentration to mean by large deviation analysis*: we use a Doob martingale argument as in [22] to show that the actual number of remaining edges Z_i well concentrates around its mean $\mathbb{E}[Z_i]$ with an exponential tail in K such that $\Pr\left(|Z_i - \mathbb{E}[Z_i]| > KC\varepsilon/2\right) \leq 2\exp\left(-c_4\varepsilon^2 K^{\frac{1}{4i+1}}\right)$ for some constant $c_4 > 0$.

□

B.1.3 Complete Decoding through Graph Expanders

From previous analyses, it has already been established that with high probability, our peeling decoder terminates with an arbitrarily small fraction of edges undecoded

$$Z_i < KC\varepsilon, \quad \forall \varepsilon > 0, \tag{50}$$

where d is the left degree. In this section, we show that the all the undecoded edges can be completely decoded if the sub-graph consisting of the remaining undecoded edges is a "good-expander". Since there are many notions of "graph expanders", we introduce the concept of graph expander with respect to the graph ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ in this paper, which is induced by subsampling.

Definition 5 (Graph Expander)**.** *A* C*-regular graph with* K *left nodes and* C *subsampling groups of* $B = \eta K$ *right nodes is called a* $(\varepsilon, 1/2, C)$*-expander if for all subsets* $\mathcal{S}$ *of left nodes with* $|\mathcal{S}| \leq \varepsilon K$*, there exists a right neighborhood in some group* c*, denoted by* $\mathcal{N}_c(\mathcal{S})$*, that satisfies* $|\mathcal{N}_c(\mathcal{S})| > |\mathcal{S}|/2$ *for some* $c \in [C]$*.*

Lemma 4 (**Graph expansion property for** $0 < \delta \leq 1/3$)**.** *In the very sparse regime* $0 < \delta \leq 1/3$*, if we use* $C \geq 3$ *groups with subsampling matrices* $\{\mathbf{M}_c\}_{c\in[C]}$ *chosen as* (42)*, then any graph from the ensemble* $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ *is a* $(\varepsilon, 1/2, C)$*-expander with probability at least* $1 - O(1/K)$ *for some sufficiently small but constant* $\varepsilon > 0$*.*

Proof. See Appendix B.7. □

Without loss of generality, let the Z_i undecoded edges be connected to a set of left nodes $\mathcal{S}$. Since each left node has degree C, it is obvious from (50) that $|\mathcal{S}| \leq K\varepsilon$ with high probability. Note that our peeling decoder fails to decode the set $\mathcal{S}$ of left nodes if and only if there are no more single-ton right nodes in the neighborhood of $\mathcal{S}$. A sufficient condition for all the right nodes in at least one group $\mathcal{N}_c(\mathcal{S})$ to have at least one single-ton is that the corresponding average degree is less than 2, which implies that $|\mathcal{S}|/|\mathcal{N}_c(\mathcal{S})| \leq 2$ and hence $|\mathcal{N}_c(\mathcal{S})| \geq |\mathcal{S}|/2$. Since we have shown in Lemma 4 that any graph from the regular ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ is a $(\varepsilon, 1/2, C)$-expander with high probability such that there is at least one group $|\mathcal{N}_c(\mathcal{S})| \geq |\mathcal{S}|/2$ for some c, there will be sufficient single-tons to peel off all the remaining edges.

B.2 Design and Analysis of a Specific Less Sparse Regime $\delta = 1 - 1/C$

From now on, we address the design and analysis for the less sparse regime $1/3 < \delta < 1$. For convenience, we start by discussing the case $\delta = 1-1/C$ where C is the number of subsampling groups. Then, we generalize our design in Section B.3 to tackle arbitrary sparsities $\delta \in (1/3, 1)$ using the basic constructions for sparsity $\delta = 1 - 1/C$. We let $t = n/C$ such that $B = 2^b$ with $b = (C-1)t$ and $B = \eta K$ for some $\eta > 0$. The subsampling matrices are chosen differently by

$$\mathbf{M}_c = \begin{bmatrix} \mathbf{I}_{(c-1)t\times(C-c)t} & \mathbf{0}_{(c-1)t\times(c-1)t} \\ \mathbf{0}_{t\times(C-c)t} & \mathbf{0}_{t\times(c-1)t} \\ \mathbf{0}_{(C-c)t\times(C-c)t} & \mathbf{I}_{(C-c)t\times(c-1)t} \end{bmatrix}, \quad c \in [C], \tag{51}$$

which freezes a t-bit segment of the time domain indices $\mathbf{m} \in \mathbb{F}_2^n$ to all zeros.

B.2.1 Random Graph Ensemble in the Less Sparse Regime $\delta = 1 - 1/C$

For convenience, we divide $\mathbf{k} = [\boldsymbol{k}_1^T, \cdots, \boldsymbol{k}_{C-1}^T, \boldsymbol{k}_C^T]^T$ into C pieces of n/C-bit segments with

$$\boldsymbol{k}_c = [k[cn/C], \cdots, k[(c-1)n/C + 1]]^T. \tag{52}$$

Then in this regime, the hash functions associated with (51) are defined as

$$\mathcal{H}_c(\mathbf{k}) = \mathbf{M}_c^T \mathbf{k} = [\boldsymbol{k}_1^T, \cdots, \boldsymbol{k}_{c-1}^T, \boldsymbol{k}_{c+1}^T, \cdots, \boldsymbol{k}_C^T]^T, \quad c \in [C], \tag{53}$$

which produces a *bit segmentation* that sifts out all but one segment $\boldsymbol{k}_c$ cyclically. Using this set of subsampling matrices (i.e. hash functions), the graph ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ in Definition 1 is also consistent with the "balls-and-bins" model. For example, when $C = 3$ and $\delta = 2/3$ such that $t = n/3$, the subsampling matrices are chosen as

$$\mathbf{M}_1 = \begin{bmatrix} \mathbf{0}_{\frac{n}{3}\times\frac{n}{3}} & \mathbf{0}_{\frac{n}{3}\times\frac{n}{3}} \\ \mathbf{I}_{\frac{n}{3}\times\frac{n}{3}} & \mathbf{0}_{\frac{n}{3}\times\frac{n}{3}} \\ \mathbf{0}_{\frac{n}{3}\times\frac{n}{3}} & \mathbf{I}_{\frac{n}{3}\times\frac{n}{3}} \end{bmatrix}, \quad \mathbf{M}_2 = \begin{bmatrix} \mathbf{I}_{\frac{n}{3}\times\frac{n}{3}} & \mathbf{0}_{\frac{n}{3}\times\frac{n}{3}} \\ \mathbf{0}_{\frac{n}{3}\times\frac{n}{3}} & \mathbf{0}_{\frac{n}{3}\times\frac{n}{3}} \\ \mathbf{0}_{\frac{n}{3}\times\frac{n}{3}} & \mathbf{I}_{\frac{n}{3}\times\frac{n}{3}} \end{bmatrix}, \quad \mathbf{M}_3 = \begin{bmatrix} \mathbf{I}_{\frac{n}{3}\times\frac{n}{3}} & \mathbf{0}_{\frac{n}{3}\times\frac{n}{3}} \\ \mathbf{0}_{\frac{n}{3}\times\frac{n}{3}} & \mathbf{I}_{\frac{n}{3}\times\frac{n}{3}} \\ \mathbf{0}_{\frac{n}{3}\times\frac{n}{3}} & \mathbf{0}_{\frac{n}{3}\times\frac{n}{3}} \end{bmatrix}. \tag{54}$$

and the bin indices corresponding to the ball $\mathbf{k}$ in the 3 groups are given by

$$\boldsymbol{j}_1 = \begin{bmatrix} k[n/3+1] \\ \vdots \\ k[n] \end{bmatrix}, \quad \boldsymbol{j}_2 = \begin{bmatrix} k[1] \\ \vdots \\ k[n/3] \\ k[2n/3+1] \\ \vdots \\ k[n] \end{bmatrix}, \quad \boldsymbol{j}_3 = \begin{bmatrix} k[1] \\ \vdots \\ k[2n/3] \end{bmatrix}. \tag{55}$$

Same as the very sparse case, since each *bit segmentation* $\boldsymbol{j}_c = \mathcal{H}_c(\mathbf{k})$ is independently and uniformly at random from $\mathbb{F}_2^n$ by Assumption 1, the bit patterns $k[i]$ for $i \in [n]$ are independently and uniformly chosen from $\{0, 1\}$ for each ball. Therefore, each left ball is thrown independently into the bins on the right, which suggests that the edges from each left node to each right node are connected independently. Further, the bin index in each group $\boldsymbol{j}_c$ contains bit segments in $\mathbf{k}$ that are uniformly distributed, and hence each ball is thrown uniformly at random to one of the B right nodes in that group. Therefore, due to the independence and uniformity of the support distribution $\mathbf{k}$, the graph ensemble is consistent with the random "balls-and-bins" model in each group.

B.2.2 Peeling Decoder over the Graph Ensemble in the Less Sparse Regime $\delta = 1 - 1/C$

The analysis of the peeling decoder in the less sparse regime depends on the graphs induced by subsampling. Note that the key difference of the graphs associated with the less sparse case from the very sparse case is that for each ball $\mathbf{k}$, although the edges are connected uniformly and independently to B bins in each group, they are no longer connected independently across different groups. However, since the graph ensemble is consistent with the "balls-and-bins" model in each group, it can be easily shown that the density evolution analysis and concentration analysis carry over to the less sparse regime based on the analysis in Section B.1. However, there are some key differences in the graph expansion properties due to the lack of independence across different groups. In this section, we focus on proving the graph expansion properties for the graph ensemble in the less sparse regime.

Lemma 5 (Graph expansion property for $\delta = 1-1/C$). *In the less sparse regime $\delta = 1-1/C$, if we use $C \geq 3$ groups with subsampling matrices $\{\mathbf{M}_c\}_{c\in[C]}$ chosen as (51) and $B = \eta K$ chosen with respect to the number of groups C according to Table 1, then any graph from the ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ is a $(\varepsilon, 1/2, C)$-expander with probability at least $1 - O\left(\frac{1}{K^{(2^C-2C)/(C-1)}}\right)$ for some sufficiently small but constant $\varepsilon > 0$.*

Proof. To show that the graph ensemble in the less sparse regime is a $(\varepsilon, 1/2, C)$ expander defined in Definition 5, we need to show that irrespective of the inter-dependence of the edges across different groups, any subset $\mathcal{S}$ of left nodes has at least one right neighborhood in one group such that $\max_{c\in[C]} |\mathcal{N}_c(\mathcal{S})| \geq |\mathcal{S}|/2$. Since it has been shown in the very sparse regime in Lemma 4 that the bottleneck event of graph expander is when the size of the set is constant $|\mathcal{S}| = O(1)$. Therefore in the following, we show that for any given subset $\mathcal{S}$ of left nodes with size $|\mathcal{S}| = s = O(1)$, their right neighborhoods will not be multi-tons with high probability.

Given an arbitrary left node with the following bit segments

$$\mathbf{k} = [k[n], \cdots, k[1]]^T = [\boldsymbol{k}_C^T, \cdots, \boldsymbol{k}_1^T]^T, \quad \boldsymbol{k}_c := [k[ct], \cdots, k[(c-1)t+1]]^T, \quad c \in [C], \tag{56}$$

its right neighbors are all multi-tons if and only if there exists at least another left node labeled $\mathbf{k}'$ in each group $c \in [C]$ such that $\mathcal{H}_c(\mathbf{k}) = \mathcal{H}_c(\mathbf{k}')$. For a pathological set $\mathcal{S}$ where $\mathcal{H}_c(\mathbf{k}) = \mathcal{H}_c(\mathbf{k}')$ for any distinct pair $\mathbf{k} \neq \mathbf{k}' \in \mathcal{S}$, the left node labels $\mathbf{k}$ and $\mathbf{k}'$ differ with each other only in one segment:

$$\boldsymbol{k}_{c_\star} \neq \boldsymbol{k}'_{c_\star}, \quad \text{for some } c_\star \in [C] \tag{57}$$

$$\boldsymbol{k}_c = \boldsymbol{k}'_c, \quad \text{for all } c \neq c_\star. \tag{58}$$

Since there are at least 2 such nodes for each group $c \in [C]$ to form multi-tons, the size of the pathological set $|\mathcal{S}| = s$ is satisfies $s \geq 2^C$. Let us consider the augmented worst case scenario where there are $s/2^{C-1}$ left nodes satisfying the pathological set requirements in (57) in one group (assuming there are only 2 such nodes in other $C-1$ groups). For all the nodes $\mathbf{k} \in \mathcal{S}$, the total possible number of left nodes that can differ in one segment $\boldsymbol{k}_c$ for some $c \in [C]$ is 2^t, and therefore the probability of having $s/2^{C-1}$ nodes from that space is $\frac{s}{2^{C-1}}/2^t$. In order for an arbitrary set of $s/2^{C-1}$ left nodes to land in the same bin on the right in all C subsampling groups, the probability can be obtained as

$$\prod_{c=1}^{C} \binom{2^t}{s/2^{C-1}} \left(\frac{s}{2^{C-1}2^t}\right)^s. \tag{59}$$

Let $\mathtt{F} = 2^t$, then the probability of this event can be obtained readily for any size s as

$$\Pr(\mathcal{S}) \leq \binom{K}{s} \prod_{c=1}^{C} \binom{\mathtt{F}}{s/2^{C-1}} \left(\frac{s}{2^{C-1}\mathtt{F}}\right)^s \tag{60}$$

$$= \binom{K}{s} \binom{\mathtt{F}}{s/2^{C-1}}^C \left(\frac{s}{2^{C-1}\mathtt{F}}\right)^{Cs}. \tag{61}$$

Using the inequality $\binom{a}{b} \leq (ae/b)^b$, we have

$$\Pr(\mathcal{S}) = O\left(\left(\frac{s}{\mathtt{F}}\right)^{\frac{s}{2^C}\left(2^C-2C\right)}\right) \tag{62}$$

Since the pathological set satisfies $s \geq 2^C$ and $K = O(\mathtt{F}^{C-1})$, we can further bound the probability as

$$\Pr(\mathcal{S}) = O\left(\frac{1}{\mathtt{F}^{2^C-2C}}\right) = O\left(\frac{1}{K^{(2^C-2C)/(C-1)}}\right). \tag{63}$$

□

B.3 Generalized Design to Arbitrary Sparsity Regime $0 \leq \delta < 1$

As of now, we have presented the subsampling design for the very sparse regime $0 < \delta \leq 1/3$ and partly for the less sparse regime $\delta = 1 - 1/C$ for $\delta = 2/3, 3/4, 4/5, 5/6, \cdots$ for all $C > 0$. However, it does not generalize to any sparsity $0 < \delta < 1$. In this section, we continue to show that using the basic constructions above, we can achieve any sparsity regime. The main idea of extending our subsampling design to an arbitrary sparsity is by the following:

- *Hash with Common Prefix*: for example, we want to design the subsampling pattern for sparsity $\delta = (1 + a)/(3+a)$ for some $a > 0$. Clearly, by varying $a \in (0, \infty)$, one can obtain an arbitrary sparsity $\delta \in (1/3, 1)$. However, we hereby note that this construction is not universal since beyond some $a_\star$, the sparse bipartite graph constructed by this design fails to work with high probability. We will show later how to determine such threshold $a_\star$ and how to achieve sparsity beyond that point. In the following, we will proceed with this example.

  We divide the bin index $\mathbf{k}$ into 4 segments $\mathbf{k} = [\boldsymbol{k}_1^T, \boldsymbol{k}_2^T, \boldsymbol{k}_3^T, \boldsymbol{k}_4^T]^T$, where $\boldsymbol{k}_1$, $\boldsymbol{k}_2$ and $\boldsymbol{k}_3$ are of equal length containing $b_c = n/(3 + a)$ bits for $c = 1, 2, 3$, while $\boldsymbol{k}_4$ contains $b_4 = an/(3 + a)$ bits. The hash function in each group is then designed with the following bit segmentation:

$$\mathcal{H}_c(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_c \\ \boldsymbol{k}_4 \end{bmatrix}, \quad c = 1, 2, 3. \tag{64}$$

  In this way, the output of the hash has b bits with

$$b = b_c + b_4 = \frac{n}{3+a} + \frac{an}{3+a} = \frac{1+a}{3+a}n = \delta n, \tag{65}$$

  and hence we have $B = 2^b = \eta K = O(N^\delta)$ for some appropriately chosen η. We refer to this generalized hash design as *common-prefix* since the hash outputs start with the same segment $\boldsymbol{k}_4$.

- *Union of Disjoint Sparse Bipartite Graphs*: using the generalized hash designed above, the sparse bipartite graph is still consistent with the balls-and-bins model, where there are $B = 2^{(b_c+b_4)}$ right nodes and K left nodes. Furthermore, since the right node of the graph is indexed by two segments $(\boldsymbol{k}_4, \boldsymbol{k}_c)$, the resulting bipartite graph can be viewed as 2^{b_4} disjoint unions of sparse bipartite graphs with $K/2^{b_4}$ left nodes and $B/2^{b_4} = 2^{b_c}$ right nodes. In other words, we have 2^{b_4} disjoint unions of graphs from the random graph ensemble $\mathcal{G}(K/2^{b_4}, 0.4073, 3, \{\mathbf{M}_c\}_{c=1,2,3})$, the decoding of which fails with probability $O(1/K/2^{b_4}) = O(1/2^{b_c})$. Therefore, by a union bound, the failure probability of peeling decoding over the bipartite graphs given by this design is

$$O\left(\frac{1}{2^{b_c}}\right) \times 2^{b_4} = O\left(\frac{1}{2^{b_c-b_4}}\right) = O\left(\frac{1}{2^{\frac{1-a}{3+a}n}}\right) = O\left(\frac{1}{2^{\frac{1+a}{3+a}n \times \left(\frac{1-a}{1+a}\right)}}\right) = O\left(\frac{1}{K^{\left(\frac{1-a}{1+a}\right)}}\right). \tag{66}$$

  Clearly, it is required that $a < a_\star = 1$ such that the failure probability approaches zero asymptotically in K. This implies a sparsity regime $\delta = (1 + a)/(3 + a) < (1 + a_\star)/(3 + a_\star) = 1/2$. Therefore, this example only works for sparsity $1/3 < \delta < 1/2$. In the following, we provide specific constructions that cover the entire sparsity regime $0 < \delta < 1$.

B.3.1 Achieving Intermediate Sparsity $0 < \delta \leq 1/3$

The design in Section B.1 can be used directly and hence we omit the discussions here.

B.3.2 Achieving Intermediate Sparsity $1/3 < \delta \leq 0.73$

Here we target sparsity $\delta = (2+a)/(6+a)$, which starts from $\delta = 1/3$ with $a = 0$ and ends at $\delta = 0.73$ with $a = 8.81$. To achieve such sparsity, we divide the bin index $\mathbf{k}$ into 7 segments

$$\mathbf{k} = [\boldsymbol{k}_1^T, \boldsymbol{k}_2^T, \boldsymbol{k}_3^T, \boldsymbol{k}_4^T, \boldsymbol{k}_5^T, \boldsymbol{k}_6^T, \boldsymbol{k}_7^T]^T, \tag{67}$$

where $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, $\boldsymbol{k}_3$, $\boldsymbol{k}_4$, $\boldsymbol{k}_5$ and $\boldsymbol{k}_6$ are of equal length containing $b_c = n/(6+a)$ bits for $c = 1, 2, \cdots, 6$, while $\boldsymbol{k}_7$ contains $b_7 = an/(6+a)$ bits. Therefore, we have $C = 6$ groups for subsampling, and the hash function in each group is designed with the following bit segmentation:

$$\mathcal{H}_1(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_2 \\ \boldsymbol{k}_3 \\ \boldsymbol{k}_7 \end{bmatrix}, \quad \mathcal{H}_2(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_3 \\ \boldsymbol{k}_7 \end{bmatrix}, \quad \mathcal{H}_3(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \boldsymbol{k}_7 \end{bmatrix} \tag{68}$$

$$\mathcal{H}_4(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_5 \\ \boldsymbol{k}_6 \\ \boldsymbol{k}_7 \end{bmatrix}, \quad \mathcal{H}_5(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_4 \\ \boldsymbol{k}_6 \\ \boldsymbol{k}_7 \end{bmatrix}, \quad \mathcal{H}_6(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_4 \\ \boldsymbol{k}_5 \\ \boldsymbol{k}_7 \end{bmatrix}. \tag{69}$$

In this way, the output of the hash has b bits with

$$b = 2b_c + b_7 = \frac{2n}{6+a} + \frac{an}{6+a} = \frac{2+a}{6+a}n = \delta n. \tag{70}$$

According to Table 1, we need to choose $B = 0.2616K$. Using the same analysis outlined before, we can show that the peeling decoder works with probability at least $1 - O(1/K)$.

B.3.3 Achieving Intermediate Sparsity $0.73 < \delta \leq 7/8$

Here we target sparsity $\delta = (3+a)/(8+a)$, which starts from $\delta = 0.73$ with $a = 10.52$ and ends at $\delta = 7/8$ with $a = 32$. To achieve such sparsity, we divide the bin index $\mathbf{k}$ into 9 segments

$$\mathbf{k} = [\boldsymbol{k}_1^T, \boldsymbol{k}_2^T, \boldsymbol{k}_3^T, \boldsymbol{k}_4^T, \boldsymbol{k}_5^T, \boldsymbol{k}_6^T, \boldsymbol{k}_7^T, \boldsymbol{k}_8^T, \boldsymbol{k}_9^T]^T, \tag{71}$$

where $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, $\boldsymbol{k}_3$, $\boldsymbol{k}_4$, $\boldsymbol{k}_5$, $\boldsymbol{k}_6$, $\boldsymbol{k}_7$ and $\boldsymbol{k}_8$ are of equal length containing $b_c = n/(8+a)$ bits for $c = 1, 2, \cdots, 8$, while $\boldsymbol{k}_9$ contains $b_9 = an/(8+a)$ bits. Therefore, we have $C = 8$ groups for subsampling, and the hash function in each group is designed with the following bit segmentation:

$$\mathcal{H}_1(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_2 \\ \boldsymbol{k}_3 \\ \boldsymbol{k}_4 \\ \boldsymbol{k}_9 \end{bmatrix}, \quad \mathcal{H}_2(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_3 \\ \boldsymbol{k}_4 \\ \boldsymbol{k}_9 \end{bmatrix}, \quad \mathcal{H}_3(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \boldsymbol{k}_4 \\ \boldsymbol{k}_9 \end{bmatrix}, \quad \mathcal{H}_4(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \boldsymbol{k}_3 \\ \boldsymbol{k}_9 \end{bmatrix} \tag{72}$$

$$\mathcal{H}_5(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_6 \\ \boldsymbol{k}_7 \\ \boldsymbol{k}_8 \\ \boldsymbol{k}_9 \end{bmatrix}, \quad \mathcal{H}_6(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_5 \\ \boldsymbol{k}_7 \\ \boldsymbol{k}_8 \\ \boldsymbol{k}_9 \end{bmatrix}, \quad \mathcal{H}_7(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_5 \\ \boldsymbol{k}_6 \\ \boldsymbol{k}_8 \\ \boldsymbol{k}_9 \end{bmatrix}, \quad \mathcal{H}_8(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_5 \\ \boldsymbol{k}_6 \\ \boldsymbol{k}_7 \\ \boldsymbol{k}_9 \end{bmatrix}. \tag{73}$$

In this way, the output of the hash has b bits with

$$b = 3b_c + b_9 = \frac{3n}{8+a} + \frac{an}{8+a} = \frac{3+a}{8+a}n = \delta n. \tag{74}$$

According to Table 1, we need to choose $B = 0.2336K$. Using the same analysis outlined before, we can show that the peeling decoder works with probability at least $1 - O(1/K^{0.9})$.

B.3.4 Achieving Intermediate Sparsity $7/8 < \delta < 1$

The sparsity index δ in the range $0.875 < \delta < 1$ can be achieved by the combination of designs proposed in the less sparse regime for increasing (but constant) number of groups C as dictated by $\delta = 1 - 1/C$. For example, we can target the sparsity setting $\delta = (7+a)/(8+a)$ starting from $\delta = 0.875$ with $a = 0$ and until $\delta = 0.99$. In this construction, we divide the bin index $\mathbf{k}$ into 9 segments

$$\mathbf{k} = [\boldsymbol{k}_1^T, \boldsymbol{k}_2^T, \boldsymbol{k}_3^T, \boldsymbol{k}_4^T, \boldsymbol{k}_5^T, \boldsymbol{k}_6^T, \boldsymbol{k}_7^T, \boldsymbol{k}_8^T, \boldsymbol{k}_9^T]^T, \tag{75}$$

where $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, $\boldsymbol{k}_3$, $\boldsymbol{k}_4$, $\boldsymbol{k}_5$, $\boldsymbol{k}_6$, $\boldsymbol{k}_7$ and $\boldsymbol{k}_8$ are of equal length containing $b_c = n/(8+a)$ bits for $c = 1, 2, \cdots, 8$, while $\boldsymbol{k}_9$ contains $b_9 = an/(8+a)$ bits. The hash function in each group is then designed with the following bit segmentation:

$$\mathcal{H}_c(\mathbf{k}) = \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_{c-1} \\ \boldsymbol{k}_{c+1} \\ \vdots \\ \boldsymbol{k}_9 \end{bmatrix}, \quad c = 1, 2, \cdots, 8. \tag{76}$$

In this way, the output of the hash has b bits with

$$b = 7b_c + b_9 = \frac{7n}{8+a} + \frac{an}{8+a} = \frac{7+a}{8+a}n = \delta n. \tag{77}$$

According to Table 1, we need to choose $B = 0.2336K$. Using the same analysis outlined before, we can show that the peeling decoder works with probability at least $1 - O(1/K)$.

B.4 Right Edge Degree Distribution

Clearly, the total number of edges is KC in the bipartite graph since there are K left nodes in the bipartite graph and each left node has degree C. Therefore, since the expected number of right nodes with degree j can be obtained as $\Pr(\text{a right node has degree } j)\, CBj$, the fraction ρ_j can be obtained as

$$\rho_j = \frac{\Pr(\text{a right node has degree } j)\, CBj}{KC} = j\eta \Pr(\text{a right node has degree } j), \tag{78}$$

where we have used $B = \eta K$ and η is the redundancy parameter. According to the "balls-and-bins" model, the degree of a right node follows the binomial distribution $B(1/\eta K, K)$ and can be well approximated by a Poisson variable as

$$\Pr(\text{a right node has degree } j) \approx \frac{(1/\eta)^j e^{-1/\eta}}{j!}. \tag{79}$$

As a result, the fraction ρ_j of edges connected to right nodes having degree j is obtained as (44).

B.5 Proof of Mean Performance

Let $Z_i^{\mathrm{e}} \in \{0, 1\}$ be the random variable denoting the presence of edge e after i iterations, thus

$$Z_i = \sum_{e=1}^{KC} Z_i^{\mathrm{e}}. \tag{80}$$

Since each edge is peeled off independently given the event $\mathcal{T}_i$, the expected number of remaining edges over cycle-free graphs can be obtained as

$$\mathbb{E}\left[Z_i|\mathcal{T}_i\right] = \sum_{e=1}^{KC} \mathbb{E}\left[Z_i^{\mathrm{e}}|\mathcal{T}_i\right] = KCp_i, \tag{81}$$

where by definition $p_i = \Pr\left(Z_i^{\mathrm{e}} = 1|\mathcal{T}_i\right)$ is the *conditional probability* of an edge in the i-th peeling iteration conditioned on the event $\mathcal{T}_i$ studied in the density evolution equation (46). We are interested in the evolution of such probability p_i. In the following, we prove that for any given $\varepsilon > 0$, there exists a finite number of iterations $i > 0$ such that $p_i \leq \varepsilon/4$, which leads to our desired result in (47).

B.6 Concentration Analysis

B.6.1 Proof of Mean Analysis on General Graphs from Ensembles

From (80), we have

$$\mathbb{E}\left[Z_i\right] = \sum_{e=1}^{KC} \mathbb{E}\left[Z_i^{\mathrm{e}}\right] = K\bar{d}\mathbb{E}\left[Z_i^{\mathrm{e}}\right]. \tag{82}$$

From basic probability laws, we have

$$\mathbb{E}\left[Z_i^{\mathrm{e}}\right] = \mathbb{E}\left[Z_i^{\mathrm{e}}|\mathcal{T}_i\right]\Pr\left(\mathcal{T}_i\right) + \mathbb{E}\left[Z_i^{\mathrm{e}}|\mathcal{T}_i^c\right]\Pr\left(\mathcal{T}_i^c\right).$$

Recall from the density evolution analysis that $\mathbb{E}\left[Z_i^{\mathrm{e}}|\mathcal{T}_i\right] = p_i$, we have

$$\Pr\left(\mathcal{T}_i\right) \leq 1, \quad \mathbb{E}\left[Z_{\mathrm{e}}|\mathcal{T}_i^c\right] \leq 1 \tag{83}$$

and therefore the following holds:

$$p_i - \Pr\left(\mathcal{T}_i^c\right) \leq \mathbb{E}\left[Z_i^{\mathrm{e}}\right] \leq p_i + \Pr\left(\mathcal{T}_i^c\right). \tag{84}$$

If the probability of a general graph not behaving like a tree can be made arbitrarily small for any $\varepsilon > 0$,

$$\Pr\left(\mathcal{T}_i^c\right) < \frac{\varepsilon}{4}, \tag{85}$$

then we can obtain the result in (48) by letting $p_i = \varepsilon/4$ in the density evolution analysis. Next, we show that (85) holds for sufficiently large K.

Lemma 6. *For any given constant $\varepsilon > 0$ and iteration $i > 0$, there exists some absolute constant $K_0 > 0$ such that*

$$\Pr\left(\mathcal{T}_i^c\right) < c_0\frac{\log^i K}{K} \tag{86}$$

for some constant $c_0 > 0$ as long as $K > K_0$.

From this lemma, we can see that for an arbitrary $\varepsilon > 0$, the result follows as long as $K > K_0$ where K_0 is the smallest constant that satisfies $K_0/\log^i K_0 > 4c_0/\varepsilon$ given ε and i. In the following we give the proof of the lemma.

Proof. Let C_i be the number of check nodes and V_i be the number of variable nodes in the neighborhood $\mathcal{N}_{\mathrm{e}}^{2i}$. Because the graph ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$ follows Poisson distributions on the right, the results in [22] are not readily applied here. Therefore, the key idea is to prove that the size of the tree neighborhood is bounded by $O(\log^i K)$ with high probability, which is intuitively clear since Poisson distributions have very light tails due to the exponential decay.

To show this, we unfold the neighborhood of an edge e up to level $i_\star$, and at each level i we upper bound the probability that the size of the tree grows larger than $O(\log^i K)$. Specifically, from the law of total probability, we upper bound the probability of not having a tree as follows for some $\kappa_1 > 0$

$$\Pr\left(\mathcal{T}_i^c\right) \leq \Pr\left(V_i > \kappa_1 \log^i K\right) + \Pr\left(C_i > \kappa_1 \log^i K\right) \tag{87}$$

$$+ \Pr\left(\mathcal{T}_i^c | V_i < \kappa_1 \log^i K, C_i < \kappa_1 \log^i K\right). \tag{88}$$

Denoting $\alpha_i = \Pr\left(V_i > \kappa_1 \log^i K\right)$, we bound the first term using the total law of probability as follows

$$\alpha_i \leq \alpha_{i-1} + \Pr\left(V_i > \kappa_1 \log^i K | V_{i-1} < \kappa_1 \log^{i-1} K\right).$$

Given $V_{i-1} < \kappa_1 \log^{i-1} K$, we have $C_i < \kappa_2 \log^{i-1} K$ at depth i for some $\kappa_2 > 0$ since the left degree of any graph from both ensembles is upper bounded by d and D respectively, which are both constants. Therefore, the second term in the above recursion can be bounded as

$$\Pr\left(V_i > \kappa_1 \log^i K | V_{i-1} < \kappa_1 \log^{i-1} K\right) \leq \Pr\left(V_i > \kappa_1 \log^i K | C_i < \kappa_2 \log^{i-1} K\right). \tag{89}$$

Now let the number of check nodes at exactly depth i be M_i and let D_i be the degrees of each of these check nodes, the right hand side can be evaluated as

$$\Pr\left(V_i > \kappa_1 \log^i K | C_i < \kappa_2 \log^{i-1} K\right) \leq \Pr\left(\sum_{i=1}^{M_i} D_i \geq \kappa_3 \log^i K\right) \tag{90}$$

for some $\kappa_3 > 0$. Since the check node degrees are Poisson variables with rate $1/\eta$ and the number of check nodes at depth i is less than the total number of check nodes up to depth i such that $M_i \leq C_i < \kappa_1 \log^i K$, then the probability can be upper bounded with $\Pr\left(D_i \geq x\right) \leq (e\lambda/x)^x$ as

$$\Pr\left(\sum_{i=1}^{M_i} D_i \geq \kappa_3 \log^i K\right) \leq \left(\frac{eM_i/\eta}{\kappa_3 \log^i K}\right)^{\kappa_3 \log^i K} \leq \left(\frac{\kappa_4}{\log K}\right)^{\kappa_3 \log^i K} \leq \frac{\kappa_5}{K} \tag{91}$$

for some $\kappa_4 > 0$ and $\kappa_5 > 0$. Therefore we have

$$\alpha_i \leq \alpha_{i-1} + \frac{\kappa_5}{K} \tag{92}$$

and thus the number of variable nodes exposed until the i-th iteration can be bounded by $\log^i K$ with high probability $\Pr\left(V_i > \kappa_1 \log^i K\right) \leq O\left(\frac{1}{K}\right)$. Similar technique can be used to show that the tail bound for the check nodes is $\Pr\left(C_i > \kappa_1 \log^i K\right) \leq O\left(\frac{1}{K}\right)$.

It has been shown that the number of nodes is well bounded by $O(\log^i K)$, now we proceed to show the tree-like neighborhood of our graph ensemble by induction. Assuming that the neighborhood $\mathcal{N}_{\mathrm{e}}^{2i}$ at the i-th iteration ($i < i_\star$) is tree-like, we prove that $\mathcal{N}_{\mathrm{e}}^{2(i+1)}$ is tree-like with high probability.

First of all, we examine the neighborhood $\mathcal{N}_{\mathrm{e}}^{2i+1}$. Assume that t additional edges have been revealed at this level without forming a cycle. The probability that the next edge from a variable node does not create a cycle is the probability that it is connected to one of the check nodes that are not already included in the tree, which is lower bound by $1 - C_{i_\star}/(\eta K)$. Therefore, given that $\mathcal{N}_{\mathrm{e}}^{2i}$ is tree-like, the probability that $\mathcal{N}_{\mathrm{e}}^{2i+1}$ is tree-like is lower bounded by

$$\left(1 - \frac{C_{i_\star}}{\eta K}\right)^{C_{i+1} - C_i}. \tag{93}$$

By similar reasoning, given that $\mathcal{N}_{\mathrm{e}}^{2i+1}$ is tree-like, the probability that $\mathcal{N}_{\mathrm{e}}^{2(i+1)}$ is tree-like is lower bounded by

$$\left(1-\frac{V_{i_\star}}{K}\right)^{V_{i+1}-V_i}. \tag{94}$$

Therefore, the probability that $\mathcal{N}_{\mathrm{e}}^{2(i+1)}$ is tree-like is lower bounded by

$$\left(1-\frac{C_{i_\star}}{\eta K}\right)^{C_{i_\star}}\left(1-\frac{V_{i_\star}}{K}\right)^{V_{i_\star}} \geq 1-\left(\frac{V_{i_\star}^2}{K}+\frac{C_{i_\star}^2}{\eta K}\right) \geq 1-O\left(\frac{\log^i K}{K}\right).$$

Therefore the probability of not being tree-like is upper bounded by

$$\Pr\left(\mathcal{T}_i^c\right) < c_0\frac{\log^i K}{K} \tag{95}$$

for some absolute constant $c_0 > 0$. □

B.6.2 Proof of Concentration to Mean by Large Deviation Analysis

Now it remains to show the concentration of Z_i around its mean $\mathbb{E}[Z_i]$. According to (80), the number of remaining edges is a sum of random variables $Z_i = \sum_{e=1}^{KC} Z_{\mathrm{e}}^{(i)}$ while summands $Z_{\mathrm{e}}^{(i)}$ are not independent with each other. Therefore, to show the concentration, we use a standard martingale argument and Azuma's inequality provided in [22] with some modifications to account for the irregular degrees of the right nodes.

Suppose that we expose the whole set of $E = KC$ edges of the graph one at a time. We let

$$Y_\ell = \mathbb{E}\left[Z_i | Z_1^{(i)}, \cdots, Z_\ell^{(i)}\right], \quad \ell = 1, \cdots, KC. \tag{96}$$

By definition, $Y_0, Y_1, \cdots, Y_{KC}$ are a Doob's martingale process, where $Y_0 = \mathbb{E}[Z_i]$ and $Y_{KC} = Z_i$. To use Azuma's inequality, it is required that $|Y_{\ell+1} - Y_\ell| \leq \Delta_\ell$ for some $\Delta_\ell > 0$. If the variable node has a regular degree d_V and the check node has a regular degree d_C, then [22] shows that $\Delta_\ell = 8(d_V d_C)^i$ with i being the number of peeling iterations. However, although we have a regular left degree $d_V = C$ in our graph ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$, the check node degree is not regular with degree d_C and therefore requires further analysis.

Proof of Finite Difference Δ_ℓ

To prove that the difference Δ_ℓ is finite for check node degrees with Poisson distributions, we first prove that the degree of all the check nodes can be upper bounded by $d_C \leq O(K^{\frac{2}{4i+1}})$ with probability[7] at least

$$c_1 K \exp\left(-c_2 K^{\frac{2}{4i+1}}\right)$$

for some constants c_1 and c_2. Let $\mathcal{B}$ be the event that at least one check node has more than $O\left(K^{\frac{2}{4i+1}}\right)$ edges, then for some $c_3 > 0$ we have

$$\Pr\left(\mathcal{B}\right) < c_3 K \exp\left(-c_2 K^{\frac{2}{4i+1}}\right). \tag{97}$$

by applying a union bound on all the $R = \eta K$ check nodes of the graphs from $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$. As a result, under the complement event $\mathcal{B}^c$, we have

$$\Delta_\ell^2 = O\left(K^{\frac{4i}{4i+1}}\right). \tag{98}$$

[7] Let X be a Poisson variable with parameter λ, then the following holds

$$\Pr\left(X > cK^{\frac{2}{4i+1}}\right) \leq \left(\frac{e\lambda}{cK^{\frac{2}{4i+1}}}\right)^{cK^{\frac{2}{4i+1}}} \leq c_1 \exp\left(-c_2 K^{\frac{2}{4i+1}}\right)$$

for some c_1 and c_2.

Large Deviation by Azuma's Inequality

For any given $\varepsilon > 0$, the tail probability of the event $Z_i > KC\varepsilon$ can be computed as

$$\begin{aligned}\Pr\left(|Z_i - \mathbb{E}[Z_i]| > \frac{KC\varepsilon}{2}\right) &\leq \Pr\left(|Z_i - \mathbb{E}[Z_i]| > \frac{KC\varepsilon}{2}\Big|\mathcal{B}^c\right) + \Pr(\mathcal{B}) \\ &\leq 2\exp\left(-\frac{K^2C^2\varepsilon^2/4}{2\sum_{\ell=1}^{KC}\Delta_\ell^2}\right) + c_3 K \exp\left(-c_2 K^{\frac{2}{4i+1}}\right) \\ &\leq 2\exp\left(-c_4\varepsilon^2 K^{\frac{1}{4i+1}}\right),\end{aligned}$$

where c_4 is some constant depending on C, η and all the other constants c_1, c_2, c_3. This concludes our proof for (49).

B.7 Proof of Expander Graphs

Given an arbitrary subset of left nodes $\mathcal{S}$ of size $|\mathcal{S}| = s$ with less than $s/2$ neighbors for all C subsampling groups. The probability of this event can be obtained readily for any size v as

$$\Pr(\mathcal{S}) \leq \binom{K}{s}\prod_{c=1}^{C}\binom{\eta K}{s/2}\left(\frac{s}{2\eta K}\right)^s \tag{99}$$

$$= \binom{K}{s}\binom{\eta K}{s/2}^C\left(\frac{s}{2\eta K}\right)^{Cs}, \tag{100}$$

where we have used the fact that the number of check nodes is ηK. Using the inequality $\binom{a}{b} \leq (ae/b)^b$, we have

$$\Pr(\mathcal{S}) \leq \left(\frac{Ke}{s}\right)^s\left(\frac{\eta Ke}{s/2}\right)^{Cs/2}\left(\frac{s}{2\eta K}\right)^{Cs} = \left(\frac{s}{K}\right)^{s\left(\frac{C}{2}-1\right)}c^s, \tag{101}$$

where $c = e(e/\eta)^{C/2}$ is some constant. Clearly, as long as $C/2 - 1 \geq 1/2$ such that $C \geq 3$, we can further bound the probability as

$$\Pr(\mathcal{S}) \leq \left(\frac{sc^2}{K}\right)^{s/2}. \tag{102}$$

It can be seen that the probability of not forming an expander depends on the size of the remaining subset $|\mathcal{S}| = s$. Now we examine two extremes with $s = O(K)$ and $s = O(1)$, and obtain the following:

$$\Pr(\mathcal{S}) = \begin{cases} e^{-K\log\left(\frac{1}{\varepsilon c^2}\right)}, & s = \varepsilon K \text{ with } \varepsilon < 1/(2c^2) \\ O\left(\frac{1}{K}\right), & s = O(1). \end{cases} \tag{103}$$

Clearly, the bottleneck event is when the graph is left with $s = O(1)$ variable nodes, which happens also with probability approaching zero asymptotically in K. Therefore, the random graphs from the ensemble are good expanders with probability at least $1 - O(1/K)$.

C Proof of Proposition 3

Given a single-ton bin with an index-value pair $(\mathbf{k}, X[\mathbf{k}])$,

$$U_p = |X[\mathbf{k}]|(-1)^{\langle \mathbf{d}_p, \mathbf{k}\rangle \oplus \mathsf{sgn}[X[\mathbf{k}]]} + W_p, \quad p \in [P], \tag{104}$$

it is clear that $\mathsf{sgn}\left[U_p\right] = \langle \mathbf{d}_p, \mathbf{k}\rangle \oplus \mathsf{sgn}\left[X[\mathbf{k}]\right] \oplus 1$ whenever the noise W_p is sufficiently large such that it crosses over $X[\mathbf{k}](-1)^{\langle \mathbf{d}_p, \mathbf{k}\rangle}$. Clearly, this is a random event and we can model it with some Bernoulli variable $Z_p \in \{0, 1\}$ with some probability p_Z

$$\mathsf{sgn}\left[U_p\right] = \langle \mathbf{d}_p, \mathbf{k}\rangle \oplus \mathsf{sgn}\left[X[\mathbf{k}]\right] \oplus Z_p. \tag{105}$$

The exact parameter p_Z of the Bernoulli random variable Z_p can be found by studying the tail events that trigger the flipping, but here for simplicity we directly upper bound it as follows

$$p_Z \leq \mathbb{P}_{\mathrm{e}} := \Pr\left(|W_p| > |X[\mathbf{k}]|\right) \leq e^{-\frac{|X[\mathbf{k}]|^2}{2N/B\sigma^2}} = e^{-\frac{\eta}{2}\mathrm{SNR}}. \tag{106}$$

D Proof of Theorem 3: Peeling Decoder using a Robust Bin Detector

Let E_{bin} be the event where the robust bin detector makes a mistake in the $O(K)$ peeling iterations. If the error probability of the robust bin detector described in Section 5 satisfies

$$\Pr\left(E_{\mathrm{bin}}\right) = O\left(\frac{1}{K}\right), \tag{107}$$

then the result directly follows from the Bayes rule:

$$\begin{aligned}\mathbb{P}_F &= \Pr\left(\mathrm{supp}\left(\widehat{\mathbf{X}}\right) \neq \mathrm{supp}\left(\mathbf{X}\right) \middle| E_{\mathrm{bin}}^c\right) \Pr\left(E_{\mathrm{bin}}^c\right) + \Pr\left(\mathrm{supp}\left(\widehat{\mathbf{X}}\right) \neq \mathrm{supp}\left(\mathbf{X}\right) \middle| E_{\mathrm{bin}}\right) \Pr\left(E_{\mathrm{bin}}\right) \\ &\leq \Pr\left(\mathrm{supp}\left(\widehat{\mathbf{X}}\right) \neq \mathrm{supp}\left(\mathbf{X}\right) \middle| E_{\mathrm{bin}}^c\right) + \Pr\left(E_{\mathrm{bin}}\right) = O\left(1/K\right),\end{aligned}$$

where the first term in the last inequality is obtained from Theorem 4 for the peeling decoder with an oracle such that the event E_0^c holds. Therefore, it remains to show that (107) holds. The main idea is to analyze the error probability of making at least an error on any bin observation, followed by a union bound on all the bin observation. Let the error event in any bin j as E_j, then we have the following union bound across ηK bin observation vectors as well as CK iterations[8]

$$\Pr\left(E_{\mathrm{bin}}\right) \leq CK \bigcup_{j=1}^{\eta K} \Pr\left(E_j\right), \tag{108}$$

where C is the left degree of the regular ensemble $\mathcal{G}(K, \eta, C, \{\mathbf{M}_c\}_{c\in[C]})$. Without loss of generality, we drop the bin index j and use a union bound over all bins such that

$$\Pr\left(E_{\mathrm{bin}}\right) \leq \eta C K^2 \Pr\left(E\right), \tag{109}$$

where $\Pr\left(E\right)$ is the error probability for an arbitrary bin. It can be seen that due to the union bounds, it is required that $\Pr\left(E\right) \leq O(1/K^3)$ such that $\Pr\left(E_{\mathrm{bin}}\right) \leq O(1/K)$.

In the following, we prove that $\Pr\left(E\right) \leq O(1/K^3)$ holds using the generic model in Proposition 2. Since there are different types of errors, thus in the following analysis $\boldsymbol{\alpha}$ in (18) is fixed as a zero-ton, single-ton or multi-ton respectively for each class of errors.

[8]The number of iterations is taken to be the worst case where at each iteration only one edge is peeled off.

Definition 6. *The error probability* $\Pr(E)$ *for an arbitrary bin can be upper bounded as*

$$\Pr(E) \leq \sum_{\mathcal{F}\in\{\mathcal{H}_Z,\mathcal{H}_M\}} \Pr(\mathcal{F} \leftarrow \mathcal{H}_S(\mathbf{k}, X[\mathbf{k}])) + \sum_{\mathcal{F}\in\{\mathcal{H}_Z,\mathcal{H}_M\}} \Pr(\mathcal{H}_S(\mathbf{k}, X[\mathbf{k}]) \leftarrow \mathcal{F}) \tag{110}$$

$$+ \Pr\left(\mathcal{H}_S(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}]) \leftarrow \mathcal{H}_S(\mathbf{k}, X[\mathbf{k}])\right) \tag{111}$$

where $\mathcal{F}$ *is either a zero-ton* $\mathcal{H}_Z$ *or a multi-ton* $\mathcal{H}_M$ *and*

1. $\Pr(\mathcal{F} \leftarrow \mathcal{H}_S(\mathbf{k}, X[\mathbf{k}]))$ *is called the missed verification rate in which the single-ton verification fails when the ground truth is in fact a single-ton* $\mathcal{H} = \mathcal{H}_S(\mathbf{k}, X[\mathbf{k}])$ *for some* $\mathbf{k} \in \mathbb{F}_2^n$ *and* $X[\mathbf{k}]$.
2. $\Pr(\mathcal{H}_S(\mathbf{k}, X[\mathbf{k}]) \leftarrow \mathcal{F})$ *is called the false verification rate in which the single-ton verification is passed for some single-ton* $\mathcal{H} = \mathcal{H}_S(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}])$ *with an index-value pair* $(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}])$ *when the ground truth is* $\mathcal{F} \in \{\mathcal{H}_Z, \mathcal{H}_M\}$.
3. $\Pr\left(\mathcal{H}_S(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}]) \leftarrow \mathcal{H}_S(\mathbf{k}, X[\mathbf{k}])\right)$ *is called the crossed verification rate in which a single-ton with a wrong index-value pair* $\widehat{\mathbf{k}} \neq \mathbf{k}, \widehat{X}[\widehat{\mathbf{k}}] \neq X[\mathbf{k}]$ *passes the single-ton verification when the ground truth is a single-ton with an index-value pair* $\mathcal{H} = \mathcal{H}_S(\mathbf{k}, X[\mathbf{k}])$ *for some* $k \neq \widehat{k}$.

The false verification rate, missed verification rate and crossed verification rate for the near-linear time and sub-linear time recovery schemes are given in the following propositions.

Proposition 5 (False Verification Rate)**.** *For any* $0 < \gamma < \mathsf{SNR}/2$*, the false verification rate for each bin hypothesis can be upper bounded as follows:*

$$\Pr\left(\mathcal{H}_S(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}]) \leftarrow \mathcal{H}_Z\right) < e^{-\frac{P_1}{4}\left(\sqrt{1+2\gamma}-1\right)^2}$$

$$\Pr\left(\mathcal{H}_S(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}]) \leftarrow \mathcal{H}_M\right) < e^{-\frac{P_1}{4}\frac{\gamma^2}{1+4\gamma}} + Ne^{-\frac{\varepsilon}{4}\left(1-\frac{2\gamma\nu^2}{\rho^2}\right)P_1},$$

where P_1 *is the number of the random offsets in the NSO-SPRIGHT and the SO-SPRIGHT algorithm.*

Proof. See Appendix E. □

Proposition 6 (Missed Verification Rate)**.** *For any* $0 < \gamma < \mathsf{SNR}/2$*, the missed verification rate for each bin hypothesis can be upper bounded as follows:*

$$\Pr(\mathcal{H}_Z \leftarrow \mathcal{H}_S(\mathbf{k}, X[\mathbf{k}])) < e^{-\frac{P_1}{4}\frac{\left(\rho^2/\nu^2-\gamma\right)^2}{1+2\rho^2/\nu^2}}$$

$$\Pr(\mathcal{H}_M \leftarrow \mathcal{H}_S(\mathbf{k}, X[\mathbf{k}])) < 2e^{-\frac{\rho^2}{2\nu^2}P_1} + \begin{cases} 2ne^{-\frac{(1-2\theta)^2}{8}P_1}, & \textit{NSO-SPRIGHT} \\ 2e^{-\frac{(\beta/\mathbb{P}_e-1)^2}{3}P_3} + 2e^{-\frac{(1-2\mathbb{P}_e)^2}{8}P_2}, & \textit{SO-SPRIGHT}. \end{cases}$$

where P_1 *is the number of random offsets in the NSO-SPRIGHT and the SO-SPRIGHT algorithm, while* P_2 *and* P_3 *are the numbers of the zero offsets and coded offsets in the SO-SPRIGHT algorithm.*

Proof. See Appendix F. □

Proposition 7 (Crossed Verification Rate)**.** *For any* $0 < \gamma < \mathsf{SNR}/2$*, the false verification rate for each bin hypothesis can be upper bounded as follows:*

$$\Pr\left(\mathcal{H}_S(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}]) \leftarrow \mathcal{H}_S(\mathbf{k}, X[\mathbf{k}])\right) < e^{-\frac{P_1}{4}\frac{\gamma^2}{1+4\gamma}} + 2Ne^{-\frac{1}{8}\left(1-\frac{\gamma\nu^2}{\rho^2}\right)^2 P_1},$$

where P_1 *is the number of random offsets in the NSO-SPRIGHT and the SO-SPRIGHT algorithm.*

Proof. See Appendix G. □

Since all the error probabilities decay exponentially with respect to $\{P_i\}_{i=1}^3$, it is now clear that if P_i is chosen as $P_i = O(n) = O(\log N)$, the probability can be bounded as $\Pr(E) = O(1/N^3)$ such that $\Pr(E_{\text{bin}}) = O(1/K)$.

E Proof of False Verification Rates in Proposition 5

The false verification events occur when the ground truth is not a single-ton, and therefore, the probabilities can be obtained using the bin observation model

$$\boldsymbol{U} = \mathbf{S}\boldsymbol{\alpha} + \boldsymbol{W} \tag{112}$$

with $\boldsymbol{\alpha}$ being a zero-ton $\boldsymbol{\alpha} = \mathbf{0}$ or a multi-ton $|\mathrm{supp}\,(\boldsymbol{\alpha})| > 1$. With a slight abuse of notation, here $\mathbf{S} \in \{\pm 1\}^{P_1 \times N}$ is the codebook associated with the P_1 fully random offsets in the NSO-SPRIGHT and SO-SPRIGHT algorithm.

E.1 Detecting a Zero-ton as a Single-ton

By definition, the probability of detecting a zero-ton as a single-ton can be upper bounded by the probability of a zero-ton failing the zero-ton verification:

$$\Pr\left(\mathcal{H}_{\mathrm{S}}(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}]) \leftarrow \mathcal{H}_{\mathrm{Z}}\right) \leq \Pr\left(\frac{1}{P_1}\left\|\boldsymbol{W}\right\|^2 \geq (1+\gamma)\nu^2\right).$$

Since $\boldsymbol{W} \sim \mathcal{N}(\mathbf{0}, \nu^2\mathbf{I})$, we can bound this probability using Lemma 11:

$$\Pr\left(\frac{1}{P_1}\left\|\boldsymbol{W}\right\|^2 \geq (1+\gamma)\nu^2\right) \leq e^{-\frac{P_1}{4}\left(\sqrt{1+2\gamma}-1\right)^2}.$$

E.2 Detecting a Multi-ton as a Single-ton

By definition, the error probability can be evaluated under the multi-ton model when it passes the single-ton verification step for some index-value pair $(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}])$

$$\Pr\left(\mathcal{H}_{\mathrm{S}}(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}]) \leftarrow \mathcal{H}_{\mathrm{M}}\right) = \Pr\left(\frac{1}{P_1}\left\|\boldsymbol{U} - \widehat{X}[\widehat{\mathbf{k}}]\mathbf{s}_{\widehat{\mathbf{k}}}\right\|^2 \leq (1+\gamma)\nu^2\right)$$

given some multi-ton observation $\boldsymbol{U} = \mathbf{S}\boldsymbol{\alpha} + \boldsymbol{W}$. Letting $\boldsymbol{g} = \mathbf{S}(\boldsymbol{\alpha} - \widehat{X}[\widehat{\mathbf{k}}]\mathbf{e}_{\widehat{\mathbf{k}}})$ and $\boldsymbol{v} = \boldsymbol{W}$, we compute this probability according to the total probability law as follows

$$\begin{aligned}
&\Pr\left(\frac{1}{P_1}\left\|\boldsymbol{g}+\boldsymbol{v}\right\|^2 \leq (1+\gamma)\nu^2\right) \\
&= \Pr\left(\frac{1}{P_1}\left\|\boldsymbol{g}+\boldsymbol{v}\right\|^2 \leq (1+\gamma)\nu^2 \Big| \frac{\left\|\boldsymbol{g}\right\|^2}{P_1} \geq 2\gamma\nu^2\right) \times \Pr\left(\frac{\left\|\boldsymbol{g}\right\|^2}{P_1} \geq 2\gamma\nu^2\right) \\
&\quad + \Pr\left(\frac{1}{P_1}\left\|\boldsymbol{g}+\boldsymbol{v}\right\|^2 \leq (1+\gamma)\nu^2 \Big| \frac{\left\|\boldsymbol{g}\right\|^2}{P_1} \leq 2\gamma\nu^2\right) \times \Pr\left(\frac{\left\|\boldsymbol{g}\right\|^2}{P_1} \leq 2\gamma\nu^2\right) \\
&\leq \Pr\left(\frac{1}{P_1}\left\|\boldsymbol{g}+\boldsymbol{v}\right\|^2 \leq (1+\gamma)\nu^2 \Big| \frac{\left\|\boldsymbol{g}\right\|^2}{P_1} \geq 2\gamma\nu^2\right) + \Pr\left(\frac{\left\|\boldsymbol{g}\right\|^2}{P_1} \leq 2\gamma\nu^2\right),
\end{aligned} \tag{113}$$

where the first term is basically the single-ton verification error rate when the multi-ton has sufficiently large energy while the second term is the probability of any multi-ton not having sufficiently large energy. In the following, we bound these two probabilities separately with exponential tails.

We start from the single-ton verification error rate when the multi-ton has sufficiently large energy, or namely $\Pr\left(\frac{1}{P_1}\left\|\boldsymbol{g}+\boldsymbol{v}\right\|^2 \leq (1+\gamma)\nu^2 \Big| \frac{\left\|\boldsymbol{g}\right\|^2}{P_1} \geq 2\gamma\nu^2\right)$. Lemma 11 can be directly used here by letting $\tau_2 = (1+\gamma)\nu^2$. Note that the first term is conditioned on the event where $\left\|\boldsymbol{g}\right\|^2/P_1 \geq 2\gamma\nu^2$, therefore the minimum normalized

non-centrality parameter can be obtained as $\nu_{\min} = \min_{\boldsymbol{g}} \|\boldsymbol{g}\|^2 / P_1 \nu^2 = 2\gamma$. Clearly, the condition for the threshold in (155) holds for Corollary 1, and thus the first term can be bounded accordingly as

$$\Pr\left(\frac{1}{P_1}\|\boldsymbol{g}+\boldsymbol{v}\|^2 \leq (1+\gamma)\nu^2 \Big| \frac{\|\boldsymbol{g}\|^2}{P_1} \geq 2\gamma\nu^2\right) \leq e^{-\frac{P_1}{4}\frac{\gamma^2}{1+4\gamma}}. \tag{114}$$

Now we examine the probability of a multi-ton not having sufficiently large energy, or namely $\Pr\left(\frac{1}{P}\|\boldsymbol{g}\|^2 \leq 2\gamma\nu^2\right)$. Letting $\boldsymbol{\beta} = \boldsymbol{\alpha} - \widehat{X}[\widehat{\mathbf{k}}]\mathbf{e}_{\widehat{\mathbf{k}}}$, we have $\boldsymbol{g} = \mathbf{S}\boldsymbol{\beta}$ and thus

$$\Pr\left(\frac{\|\boldsymbol{g}\|^2}{P_1} \leq 2\gamma\nu^2\right) = \Pr\left(\frac{\|\mathbf{S}\boldsymbol{\beta}\|^2}{P_1} \leq 2\gamma\nu^2\right). \tag{115}$$

Denoting the support of $\mathcal{L} := \operatorname{supp}(\boldsymbol{\beta})$, we bound this probability with respect to the following two multi-ton scenarios:

- $|\mathcal{L}| = L = O(1)$ where the multi-ton size is a constant. Note that $\|\mathbf{S}\boldsymbol{\beta}\|^2 = \boldsymbol{\beta}_{\mathcal{L}}^T \mathbf{S}_{\mathcal{L}}^T \mathbf{S}_{\mathcal{L}} \boldsymbol{\beta}_{\mathcal{L}}$ where $\mathbf{S}_{\mathcal{L}}$ is the sub-matrix consisting of the columns $\mathbf{k} \in \mathcal{L}$ and $\boldsymbol{\beta}_{\mathcal{L}}$ is the sub-vector containing the elements in the set $\mathbf{k} \in \mathcal{L}$. Then, we have

$$\lambda_{\min}\left(\mathbf{S}_{\mathcal{L}}^T\mathbf{S}_{\mathcal{L}}\right)\|\boldsymbol{\beta}_{\mathcal{L}}\|^2 \leq \boldsymbol{\beta}_{\mathcal{L}}^T\mathbf{S}_{\mathcal{L}}^T\mathbf{S}_{\mathcal{L}}\boldsymbol{\beta}_{\mathcal{L}} \leq \lambda_{\max}\left(\mathbf{S}_{\mathcal{L}}^T\mathbf{S}_{\mathcal{L}}\right)\|\boldsymbol{\beta}_{\mathcal{L}}\|^2. \tag{116}$$

  Using $\|\boldsymbol{\beta}_{\mathcal{L}}\|^2 \geq L\rho^2$, the probability can be bounded as

$$\Pr\left(\frac{\|\boldsymbol{g}\|^2}{P_1} \leq 2\gamma\nu^2\right) = \Pr\left(\frac{\|\mathbf{S}_{\mathcal{L}}\boldsymbol{\beta}_{\mathcal{L}}\|^2}{P_1} \leq 2\gamma\nu^2\right) \tag{117}$$

$$\leq \Pr\left(\lambda_{\min}\left(\frac{1}{P_1}\mathbf{S}_{\mathcal{L}}^T\mathbf{S}_{\mathcal{L}}\right) \leq \frac{2\gamma\nu^2}{\|\boldsymbol{\beta}_{\mathcal{L}}\|^2}\right) \tag{118}$$

$$= \Pr\left(\lambda_{\min}\left(\frac{1}{P_1}\mathbf{S}_{\mathcal{L}}^T\mathbf{S}_{\mathcal{L}}\right) \leq \frac{2\gamma\nu^2}{L\rho^2}\right). \tag{119}$$

Lemma 7. *Denote the mutual coherence of the codebook* $\mathbf{S}$ *by* $\mu := \max_{\mathbf{k}\neq\mathbf{m}} \frac{1}{P_1}\left|\mathbf{s}_{\mathbf{k}}^T\mathbf{s}_{\mathbf{m}}\right|$. *Then for some given* $\mu_0 > 0$, *we have* $\Pr(\mu \geq \mu_0) \leq 2Ne^{-\frac{\mu_0^2}{2}P_1}$.

Proof. Since $\mathbf{S}$ contains i.i.d. Rademacher entries, the result follows by a simple Hoeffding bound. □

According to the Gershgorin Circle Theorem

$$\lambda_{\min}\left(\frac{1}{P_1}\mathbf{S}_{\mathcal{L}}^T\mathbf{S}_{\mathcal{L}}\right) \geq 1 - L\mu \tag{120}$$

we have the following bound

$$\Pr\left(\lambda_{\min}\left(\frac{1}{P_1}\mathbf{S}_{\mathcal{L}}^T\mathbf{S}_{\mathcal{L}}\right) \leq \frac{2\gamma\nu^2}{L\rho^2}\right) \leq \Pr\left(1 - L\mu \leq \frac{2\gamma\nu^2}{L\rho^2}\right) \tag{121}$$

$$= \Pr\left(\mu \geq \frac{1}{L}\left(1 - \frac{2\gamma\nu^2}{L\rho^2}\right)\right). \tag{122}$$

By letting $\mu_0 = \frac{1}{L}\left(1 - \frac{2\gamma\nu^2}{L\rho^2}\right)$, we can upper bound this probability using Lemma 7 as

$$\Pr\left(\frac{\|\boldsymbol{g}\|^2}{P_1} \leq 2\gamma\nu^2\right) \leq \Pr\left(\mu \geq \frac{1}{L}\left(1 - \frac{2\gamma\nu^2}{L\rho^2}\right)\right) \leq 2Ne^{-\frac{1}{2L^2}\left(1-\frac{2\gamma\nu^2}{L\rho^2}\right)^2 P_1}, \tag{123}$$

which holds if $\gamma < L\rho^2/2\nu^2$.

- $|\mathcal{L}| = L = \omega(1)$ where the multi-ton size is not a constant and grows asymptotically with respect to K. As a result, the vector of random variables $\boldsymbol{g} = \mathbf{S}_{\mathcal{L}}\boldsymbol{\beta}_{\mathcal{L}}$ becomes asymptotically Gaussian due to the central limit theorem with zero mean and a covariance

$$\mathbb{E}\left[\boldsymbol{g}\boldsymbol{g}^T\right] = \mathbb{E}\left[\mathbf{S}_{\mathcal{L}}\boldsymbol{\beta}_{\mathcal{L}}\boldsymbol{\beta}_{\mathcal{L}}^T\mathbf{S}_{\mathcal{L}}^T\right] = L\rho^2\mathbf{I}. \tag{124}$$

  Therefore, from Lemma 11 and Corollary 1 we have

$$\Pr\left(\frac{\|\boldsymbol{g}\|^2}{P} \leq 2\gamma\nu^2\right) \leq e^{-\frac{1}{4}\left(1-\frac{2\gamma\nu^2}{L\rho^2}\right)P_1},$$

  which holds if $\gamma < L\rho^2/2\nu^2$.

Finally, as long as $0 < \gamma < \rho^2/2\nu^2$, for any multi-ton there exists some constant $\varepsilon > 0$ such that

$$\Pr\left(\frac{\|\boldsymbol{g}\|^2}{P_1} \leq 2\gamma\nu^2\right) \leq Ne^{-\frac{\varepsilon}{4}\left(1-\frac{2\gamma\nu^2}{\rho^2}\right)P_1}.$$

F Proof of Missed Verification Rates in Proposition 6

The missed verification events occur when the ground truth is a single-ton, and therefore, the probabilities is obtained using the bin observation model with some index-value pair $(\mathbf{k}, X[\mathbf{k}])$

$$\boldsymbol{U} = X[\mathbf{k}]\mathbf{s}_{\mathbf{k}} + \boldsymbol{W}. \tag{125}$$

With a slight abuse of notation, here $\mathbf{S}$ is the codebook associated with the fully random offsets in our designs.

F.1 Detecting a Single-ton as a Zero-ton

By definition, the probability of detecting a single-ton as a zero-ton can be upper bounded by the probability of a single-ton passing the zero-ton verification:

$$\Pr\left(\mathcal{H}_{\mathrm{Z}} \leftarrow \mathcal{H}_{\mathrm{S}}(\mathbf{k}, X[\mathbf{k}])\right) \leq \Pr\left(\frac{1}{P}\left\|X[\mathbf{k}]\mathbf{s}_{\mathbf{k}} + \boldsymbol{W}\right\|^2 \leq (1+\gamma)\nu^2\right).$$

Since $\boldsymbol{W} \sim \mathcal{N}(\mathbf{0}, \nu^2\mathbf{I})$, we can bound this probability using Lemma 11 by letting $\boldsymbol{g} = X[\mathbf{k}]\mathbf{s}_{\mathbf{k}}$ and $\boldsymbol{v} = \boldsymbol{W}$:

$$\Pr\left(\frac{1}{P}\left\|X[\mathbf{k}]\mathbf{s}_{\mathbf{k}} + \boldsymbol{W}\right\|^2 \leq (1+\gamma)\nu^2\right) \leq e^{-\frac{P_1}{4}\frac{\left(\rho^2/\nu^2-\gamma\right)^2}{1+2\rho^2/\nu^2}},$$

which holds as long as $\gamma < \rho^2/\nu^2$.

F.2 Detecting a Single-ton as a Multi-ton

By definition, the error probability can be evaluated under the single-ton model when it fails the single-ton verification step for some index-value pair $(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}])$

$$\Pr\left(\mathcal{H}_{\mathrm{M}} \leftarrow \mathcal{H}_{\mathrm{S}}(\mathbf{k}, X[\mathbf{k}])\right) = \Pr\left(\frac{1}{P}\left\|\boldsymbol{U} - \widehat{X}[\widehat{\mathbf{k}}]\mathbf{s}_{\widehat{\mathbf{k}}}\right\|^2 \geq (1+\gamma)\nu^2\right)$$

given some single-ton observation $\boldsymbol{U} = X[\mathbf{k}]\mathbf{s}_{\mathbf{k}} + \boldsymbol{W}$. Since the estimated index-value pair $(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}])$ may or may not be correct, the above probability can be bounded as:

$$\begin{aligned}
&\Pr\left(\frac{1}{P_1}\left\|\boldsymbol{U} - \widehat{X}[\widehat{\mathbf{k}}]\mathbf{s}_{\widehat{\mathbf{k}}}\right\|^2 \geq (1+\gamma)\nu^2\right) \\
&= \Pr\left(\frac{1}{P_1}\left\|\boldsymbol{U} - \widehat{X}[\widehat{\mathbf{k}}]\mathbf{s}_{\widehat{\mathbf{k}}}\right\|^2 \geq (1+\gamma)\nu^2 \middle| \widehat{X}[\widehat{\mathbf{k}}] \neq X[\mathbf{k}] \text{ or } \widehat{\mathbf{k}} \neq \mathbf{k}\right) \Pr\left(\widehat{X}[\widehat{\mathbf{k}}] \neq X[\mathbf{k}] \text{ or } \widehat{\mathbf{k}} \neq \mathbf{k}\right) \\
&\quad + \Pr\left(\frac{1}{P_1}\left\|\boldsymbol{U} - \widehat{X}[\widehat{\mathbf{k}}]\mathbf{s}_{\widehat{\mathbf{k}}}\right\|^2 \geq (1+\gamma)\nu^2 \middle| \widehat{X}[\widehat{\mathbf{k}}] = X[\mathbf{k}] \text{ and } \widehat{\mathbf{k}} = \mathbf{k}\right) \Pr\left(\widehat{X}[\widehat{\mathbf{k}}] = X[\mathbf{k}] \text{ and } \widehat{\mathbf{k}} = \mathbf{k}\right) \\
&\leq \Pr\left(\widehat{X}[\widehat{\mathbf{k}}] \neq X[\mathbf{k}] \text{ or } \widehat{\mathbf{k}} \neq \mathbf{k}\right) + \Pr\left(\frac{1}{P_1}\left\|\boldsymbol{U} - \widehat{X}[\widehat{\mathbf{k}}]\mathbf{s}_{\widehat{\mathbf{k}}}\right\|^2 \geq (1+\gamma)\nu^2 \middle| \widehat{X}[\widehat{\mathbf{k}}] = X[\mathbf{k}] \text{ and } \widehat{\mathbf{k}} = \mathbf{k}\right).
\end{aligned}$$

It is clear that

$$\Pr\left(\frac{1}{P_1}\left\|\boldsymbol{U} - \widehat{X}[\widehat{\mathbf{k}}]\mathbf{s}_{\widehat{\mathbf{k}}}\right\|^2 \geq (1+\gamma)\nu^2 \middle| \widehat{X}[\widehat{\mathbf{k}}] = X[\mathbf{k}] \text{ and } \widehat{\mathbf{k}} = \mathbf{k}\right) \tag{126}$$

$$= \Pr\left(\frac{1}{P_1}\|\boldsymbol{W}\|^2 \geq (1+\gamma)\nu^2\right) \leq e^{-\frac{P_1}{4}\left(\sqrt{1+2\gamma}-1\right)^2}, \tag{127}$$

therefore we focus on bounding the first term $\Pr\left(\widehat{X}[\widehat{\mathbf{k}}] \neq X[\mathbf{k}] \text{ or } \widehat{\mathbf{k}} \neq \mathbf{k}\right)$. From basic probability laws we have

$$\Pr\left(\widehat{X}[\widehat{\mathbf{k}}] \neq X[\mathbf{k}] \text{ or } \widehat{\mathbf{k}} \neq \mathbf{k}\right) \tag{128}$$

$$\leq \Pr\left(\widehat{X}[\widehat{\mathbf{k}}] \neq X[\mathbf{k}]\right) + \Pr\left(\widehat{\mathbf{k}} \neq \mathbf{k}\right) \tag{129}$$

$$= \Pr\left(\widehat{X}[\widehat{\mathbf{k}}] \neq X[\mathbf{k}] \middle| \widehat{\mathbf{k}} \neq \mathbf{k}\right) \Pr\left(\widehat{\mathbf{k}} \neq \mathbf{k}\right) + \Pr\left(\widehat{X}[\widehat{\mathbf{k}}] \neq X[\mathbf{k}] \middle| \widehat{\mathbf{k}} = \mathbf{k}\right) \Pr\left(\widehat{\mathbf{k}} = \mathbf{k}\right) + \Pr\left(\widehat{\mathbf{k}} \neq \mathbf{k}\right) \tag{130}$$

$$\leq \Pr\left(\widehat{X}[\widehat{\mathbf{k}}] \neq X[\mathbf{k}] \middle| \widehat{\mathbf{k}} = \mathbf{k}\right) + 2\Pr\left(\widehat{\mathbf{k}} \neq \mathbf{k}\right). \tag{131}$$

The first term is the detection error probability of a BPSK signal with amplitudes $\pm\rho$, and can be bounded as

$$\Pr\left(\widehat{X}[\widehat{\mathbf{k}}] \neq X[\mathbf{k}] \middle| \widehat{\mathbf{k}} = \mathbf{k}\right) \leq 2e^{-\frac{\rho^2}{2\nu^2}P_1}. \tag{132}$$

Since the second term $\Pr\left(\widehat{\mathbf{k}} \neq \mathbf{k}\right)$ is essentially the error probability of the single-ton search, we prove the following lemmas for different bin detection schemes.

Lemma 8 (**Single-ton Search Error Probability of the NSO-SPRIGHT Algorithm).** *The single-ton search error probability of the NSO-SPRIGHT algorithm is upper bounded as*

$$\Pr\left(\widehat{\mathbf{k}} \neq \mathbf{k}\right) \leq ne^{-\frac{(1-2\theta)^2}{8}P_1} \tag{133}$$

where P_1 is the number of random offsets in the NSO-SPRIGHT design.

Proof. See Appendix H.1. □

Lemma 9 (**Single-ton Search Error Probability of the SO-SPRIGHT Algorithm).** *The single-ton search error probability of the SO-SPRIGHT algorithm is upper bounded as*

$$\Pr\left(\widehat{\mathbf{k}} \neq \mathbf{k}\right) \leq e^{-\frac{\left(\beta/\mathbb{P}_{\mathrm{e}}-1\right)^2}{3}P_3} + e^{-\frac{(1-2\mathbb{P}_{\mathrm{e}})^2}{8}P_2}, \tag{134}$$

where P_1 is the number of the coded offsets $\mathbf{G}$ and P_2 is the number of zero offsets in the SO-SPRIGHT design.

Proof. See Appendix H.2. □

G Proof of Crossed Verification Rates in Proposition 7

A crossed verification implies that some wrong index-value pair $(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}])$ passes the single-ton verification

$$\begin{aligned}\Pr\left(\mathcal{H}_{\mathrm{S}}(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}]) \leftarrow \mathcal{H}_{\mathrm{S}}(\mathbf{k}, \widehat{X}[\mathbf{k}])\right) &= \Pr\left(\frac{1}{P_1}\left\|\boldsymbol{U} - \widehat{X}[\widehat{\mathbf{k}}]\mathbf{s}_{\widehat{k}}\right\|^2 \leq (1+\gamma)\nu^2\right) \\ &= \Pr\left(\frac{1}{P_1}\left\|X[\mathbf{k}]\mathbf{s}_{\mathbf{k}} - \widehat{X}[\widehat{\mathbf{k}}]\mathbf{s}_{\widehat{\mathbf{k}}} + \boldsymbol{W}\right\|^2 \leq (1+\gamma)\nu^2\right).\end{aligned}$$

Letting $\boldsymbol{g} = X[\mathbf{k}]\mathbf{s}_{\mathbf{k}} - \widehat{X}[\widehat{\mathbf{k}}]\mathbf{s}_{\widehat{\mathbf{k}}}$, this can be re-written as

$$\Pr\left(\mathcal{H}_{\mathrm{S}}(\widehat{\mathbf{k}}, \widehat{X}[\widehat{\mathbf{k}}]) \leftarrow \mathcal{H}_{\mathrm{S}}(\mathbf{k}, \widehat{X}[\mathbf{k}])\right) = \Pr\left(\frac{1}{P_1}\left\|\boldsymbol{g} + \boldsymbol{W}\right\|^2 \leq (1+\gamma)\nu^2\right).$$

Similar to (113), we have

$$\Pr\left(\frac{1}{P_1}\left\|\boldsymbol{g} + \boldsymbol{W}\right\|^2 \leq (1+\gamma)\nu^2\right) \leq \Pr\left(\frac{1}{P_1}\left\|\boldsymbol{g} + \boldsymbol{W}\right\|^2 \leq (1+\gamma)\nu^2 \Big| \frac{\|\boldsymbol{g}\|^2}{P_1} \geq 2\gamma\nu^2\right) + \Pr\left(\frac{\|\boldsymbol{g}\|^2}{P_1} \leq 2\gamma\nu^2\right).$$

Similar to (114), the first term can be bounded as

$$\Pr\left(\frac{1}{P_1}\left\|\boldsymbol{g} + \boldsymbol{W}\right\|^2 \leq (1+\gamma)\nu^2 \Big| \frac{\|\boldsymbol{g}\|^2}{P_1} \geq 2\gamma\nu^2\right) \leq e^{-\frac{P_1}{4}\frac{\gamma^2}{1+4\gamma}}. \tag{135}$$

Finally, similar to (136) with $L = 2$, the second term $\Pr\left(\frac{\|\boldsymbol{g}\|^2}{P_1} \leq 2\gamma\nu^2\right)$ can be bounded as

$$\Pr\left(\frac{\|\boldsymbol{g}\|^2}{P_1} \leq 2\gamma\nu^2\right) \leq 2Ne^{-\frac{1}{8}\left(1-\frac{\gamma\nu^2}{\rho^2}\right)^2 P_1}, \tag{136}$$

H Proof of Single-ton Search Error Probability in Lemma 8 and 9

H.1 Single-ton Search in the NSO-SPRIGHT Algorithm

From the MLE in (31), the error probability of the single-ton search for the q-th bit of $\mathbf{k}$ is

$$\Pr\left(\widehat{k}[q] \neq k[q]\right) = \Pr\left(\sum_{p=1}^{P_1} \mathsf{sgn}\left[U_{p,q}\right] \oplus \mathsf{sgn}\left[U_p\right] \oplus \widehat{k}[q] < \sum_{p=1}^{P_1} \mathsf{sgn}\left[U_{p,q}\right] \oplus \mathsf{sgn}\left[U_p\right] \oplus k[q]\right). \tag{137}$$

Recall that $\mathsf{sgn}\left[U_{p,q}\right] \oplus \mathsf{sgn}\left[U_p\right] = k[q] \oplus Z'_{p,q}$ in (29) where $Z'_{p,q}$ is a Bernoulli variable with probability $\theta = 2\mathbb{P}_{\mathrm{e}}(1-\mathbb{P}_{\mathrm{e}})$. Therefore, we have

$$\Pr\left(\widehat{k}[q] \neq k[q]\right) = \Pr\left(\sum_{p=1}^{P_1} k[q] \oplus \widehat{k}[q] \oplus Z'_{p,q} < \sum_{p=1}^{P_1} k[q] \oplus k[q] \oplus Z'_{p,q}\right) \tag{138}$$

$$= \Pr\left(\sum_{p=1}^{P_1} 1 \oplus Z'_{p,q} < \sum_{p=1}^{P_1} Z'_{p,q}\right). \tag{139}$$

Noticing that $\sum_{p=1}^{P_1} 1 \oplus Z'_{p,q} = P_1 - \sum_{p=1}^{P_1} Z'_{p,q}$, we have

$$\Pr\left(\widehat{k}[q] \neq k[q]\right) = \Pr\left(\sum_{p=1}^{P_1} Z'_{p,q} > P_1/2\right) \leq e^{-\frac{(1-2\theta)^2}{8}P_1}, \tag{140}$$

where the inequality follows from the Hoeffding bound. By union bounding over all n bits, we have

$$\Pr\left(\widehat{\mathbf{k}} \neq \mathbf{k}\right) \leq n e^{-\frac{(1-2\theta)^2}{8} P_1}. \tag{141}$$

H.2 Single-ton Search in the SO-SPRIGHT Algorithm

In the general setting, the index is decoded after obtaining the sign $\widehat{\mathsf{sgn}}\left[X[\mathbf{k}]\right]$. Therefore, we have

$$\Pr\left(\widehat{\mathbf{k}} \neq \mathbf{k}\right) = \Pr\left(\widehat{\mathbf{k}} \neq \mathbf{k} \middle| \widehat{\mathsf{sgn}}\left[X[\mathbf{k}]\right] = \mathsf{sgn}\left[X[\mathbf{k}]\right]\right) \Pr\left(\widehat{\mathsf{sgn}}\left[X[\mathbf{k}]\right] = \mathsf{sgn}\left[X[\mathbf{k}]\right]\right) \tag{142}$$

$$+ \Pr\left(\widehat{\mathbf{k}} \neq \mathbf{k} \middle| \widehat{\mathsf{sgn}}\left[X[\mathbf{k}]\right] \neq \mathsf{sgn}\left[X[\mathbf{k}]\right]\right) \Pr\left(\widehat{\mathsf{sgn}}\left[X[\mathbf{k}]\right] \neq \mathsf{sgn}\left[X[\mathbf{k}]\right]\right) \tag{143}$$

$$\leq \Pr\left(\widehat{\mathbf{k}} \neq \mathbf{k} \middle| \widehat{\mathsf{sgn}}\left[X[\mathbf{k}]\right]\right) + \Pr\left(\widehat{\mathsf{sgn}}\left[X[\mathbf{k}]\right] \neq \mathsf{sgn}\left[X[\mathbf{k}]\right]\right). \tag{144}$$

If the codebook $\mathbf{G}$ with block length $P_3 = O(n)$ has a minimum distance of βP_3 such that $\beta > \mathbb{P}_\mathrm{e}$, the $\mathbf{k}$ fails to be decoded when there are more than βP_3 sign flips. This can be bounded for the BSC($\mathbb{P}_\mathrm{e}$) by the Chern-off bound

$$\Pr\left(\widehat{\mathbf{k}} \neq \mathbf{k} \middle| \widehat{\mathsf{sgn}}\left[X[\mathbf{k}]\right]\right) \leq e^{-\frac{(\beta/\mathbb{P}_\mathrm{e}-1)^2}{3} P_3}. \tag{145}$$

Since the sign is obtained from P_2 sign observations through a majority test if $\mathbb{P}_\mathrm{e} < 1/2$ and a minority test if $\mathbb{P}_\mathrm{e} > 1/2$, the error in mistaking the sign can be bounded similarly to (140) as

$$\Pr\left(\widehat{\mathsf{sgn}}\left[X[\mathbf{k}]\right] \neq \mathsf{sgn}\left[X[\mathbf{k}]\right]\right) \leq e^{-\frac{(1-2\mathbb{P}_\mathrm{e})^2}{8} P_2}. \tag{146}$$

I Tail Bounds

Here we derive some tail bounds that are useful in our analysis.

Lemma 10 (Non-central Chi-Square Tail Bounds in [25])**.** *Let $Z \sim \chi^2_D$ be a non-central chi square variable with D degrees of freedom and non-centrality parameter $\theta \geq 0$. Then for all $z \geq 0$, the following tail bounds hold:*

$$\Pr\left(Z \geq (D+\theta) + 2\sqrt{(D+2\theta)z} + 2z\right) \leq \exp(-z)$$

$$\Pr\left(Z \leq (D+\theta) - 2\sqrt{(D+2\theta)z}\right) \leq \exp(-z)$$

Lemma 11. *Given $\boldsymbol{g} = [g[0], \cdots, g[P-1]]^T$ and a vector $\boldsymbol{v} = [v[0], \cdots, v[P-1]]^T$ with i.i.d. Gaussian variates $v[p] \sim \mathcal{N}(0, \nu^2)$ for all $p \in [P]$, the following tail bound holds:*

$$\Pr\left(\frac{1}{P}\left\|\boldsymbol{g}+\boldsymbol{v}\right\|^2 \geq \tau_1\right) \leq e^{-\frac{P}{4}\left(\sqrt{2\tau_1/\nu^2 - 1} - \sqrt{1+2\theta_0}\right)^2} \tag{147}$$

$$\Pr\left(\frac{1}{P}\left\|\boldsymbol{g}+\boldsymbol{v}\right\|^2 \leq \tau_2\right) \leq e^{-\frac{P}{4}\frac{\left(1+\theta_0-\tau_2/\nu^2\right)^2}{1+2\theta_0}} \tag{148}$$

for any τ_1 and τ_2 that satisfy

$$\tau_1 \geq \nu^2(1+\theta_0), \quad \tau_2 \leq \nu^2(1+\theta_0), \tag{149}$$

where θ_0 is the normalized non-centrality parameter given by

$$\theta_0 := \frac{\left\|\boldsymbol{g}\right\|^2}{P\nu^2}. \tag{150}$$

Proof. The quantity $\|\boldsymbol{g}+\boldsymbol{v}\|^2$ can be written element-wise as

$$\|\boldsymbol{g}+\boldsymbol{v}\|^2 = \sum_{p=0}^{P-1} (s[p]+v[p])^2 \tag{151}$$

where each summand is a normal random variable with mean $u[p]$ and variance ν^2. Therefore, according to the definition of non-central chi-square variables, the quantity

$$\frac{\|\boldsymbol{g}+\boldsymbol{v}\|^2}{\nu^2} \sim \chi_P^2 \tag{152}$$

is a non-central χ^2 random variable of P degrees of freedom with a non-centrality parameter

$$\theta = \sum_{p=0}^{P-1} \frac{|s[p]|^2}{\nu^2} = \frac{\|\boldsymbol{g}\|^2}{\nu^2}. \tag{153}$$

For notational convenience, we use the normalized non-centrality parameter θ_0 in (150) such that $\theta = P\theta_0$. Without loss of generality, let the thresholds τ_1 and τ_2 take the following form with respect to z_1 and z_2:

$$\begin{aligned}
\tau_1 &= \frac{\nu^2}{P}\left[(P+P\theta_0) + 2\sqrt{(P+2P\theta_0)z_1} + 2z_1\right] \\
\tau_2 &= \frac{\nu^2}{P}\left[(P+P\theta_0) - 2\sqrt{(P+2P\theta_0)z_2}\right],
\end{aligned}$$

then the tail bounds in Lemma 10 can be obtained easily with respect to z_1 and z_2. Using (153), the corresponding z_1 and z_2 can be solved as

$$\begin{aligned}
z_1 &= \frac{P}{4}\left(\sqrt{2\tau_1/\nu^2 - 1} - \sqrt{1+2\theta_0}\right)^2 \\
z_2 &= \frac{P}{4}\frac{\left(1+\theta_0-\tau_2/\nu^2\right)^2}{1+2\theta_0}
\end{aligned}$$

as long as the thresholds τ_1 and τ_2 satisfy (149). Thus according to Lemma 10, we have the tail bounds in (147). □

Corollary 1. *Suppose that the normalized non-centrality parameter θ_0 in Lemma 11 is bounded between*

$$0 \le \theta_{\min} \le \theta_0 \le \theta_{\max}, \tag{154}$$

then the following worst case tail bounds hold:

$$\begin{aligned}
\Pr\left(\frac{1}{P}\|\boldsymbol{g}+\boldsymbol{v}\|^2 \ge \tau_1\right) &\le e^{-\frac{P}{4}\left(\sqrt{2\tau_1/\nu^2-1}-\sqrt{1+2\theta_{\max}}\right)^2} \\
\Pr\left(\frac{1}{P}\|\boldsymbol{g}+\boldsymbol{v}\|^2 \le \tau_2\right) &\le e^{-\frac{P}{4}\frac{\left(1+\theta_{\min}-\tau_2/\nu^2\right)^2}{1+2\theta_{\min}}}
\end{aligned}$$

for any τ_1 and τ_2 that satisfy

$$\tau_1 \ge \nu^2(1+\theta_{\max}), \quad \tau_2 \le \nu^2(1+\theta_{\min}). \tag{155}$$

Proof. The first tail bound can be easily obtained since $\tau_1 \ge \nu^2(1+\theta_{\max})$, the exponent is monotonically decreasing with respect to θ_0, and therefore substituting it with $\theta_{\max}$ leads to an upper bound.

The second tail bound depends on the monotonicity with respect to θ_0. The tail bound is monotonic with respect to the exponent, so in the following we examine the monotonicity of the exponent with respect to θ_0. The exponent can be re-written as a form of the $x + 1/x$ function:

$$\frac{\left(1+\theta_{\min}-\tau_2/\nu^2\right)^2}{1+2\theta_{\min}} = \left(\theta_0+\frac{1}{2}\right) + \frac{\left(\frac{1}{2}-\frac{\tau_2}{\nu^2}\right)^2}{\left(\theta_0+\frac{1}{2}\right)} + 2\left(\frac{1}{2}-\frac{\tau_2}{\nu^2}\right), \tag{156}$$

which has a minimum at

$$\theta_0^\star = \left|\frac{1}{2}-\frac{\tau_2}{\nu^2}\right| - \frac{1}{2}, \tag{157}$$

and monotonically increasing for any $\theta_0 > \theta_0^\star$. Now it remains to see whether $\theta_0^\star$ is within the interval $[\theta_{\min}, \theta_{\max}]$, which needs to be discussed separately depending on the choice of τ_2:

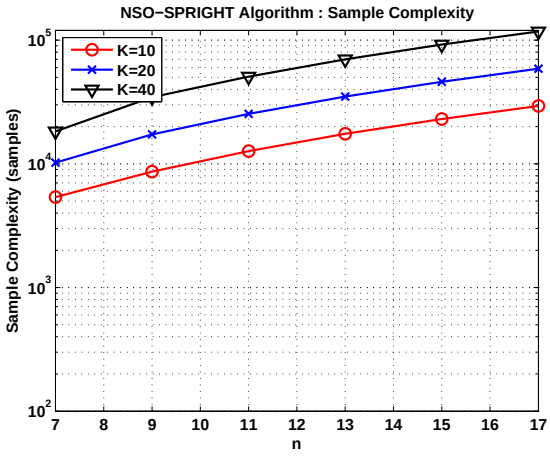
1. $\nu^2/2 \le \tau_2 \le \nu^2(1+\theta_{\min})$: in this case, we have

$$\theta_0^\star = \frac{\tau_2}{\nu^2} - 1 \le \theta_{\min}. \tag{158}$$

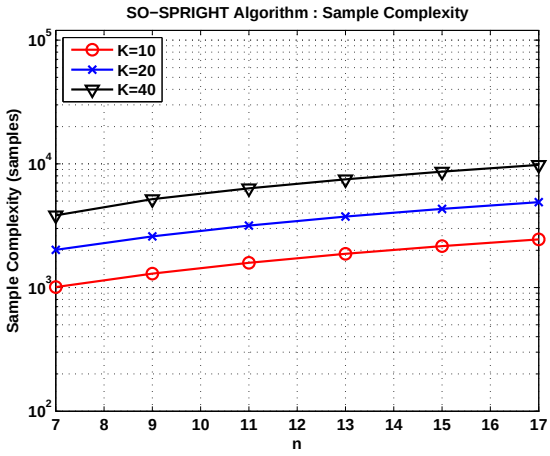
2. $0 < \tau_2 < \nu^2/2$: in this case, we have

$$\theta_0^\star = -\frac{\tau_2}{\nu^2} \le 0 \le \theta_{\min}. \tag{159}$$

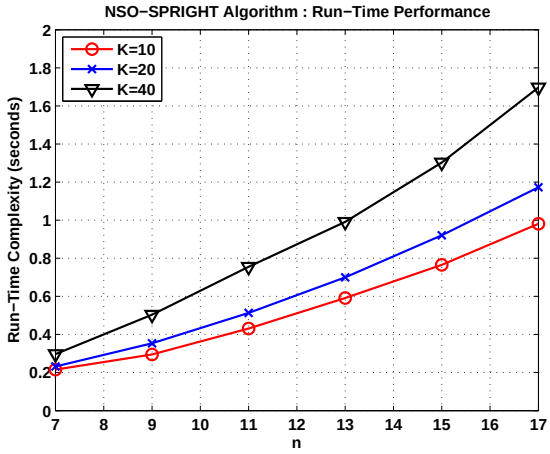
Therefore, it has been shown that as long as τ_2 satisfies (155), the exponent is monotonically increasing with respect to $\theta_0 \in [\theta_{\min}, \theta_{\max}]$ and therefore the minimum exponent is achieved by substituting θ_0 with $\theta_{\min}$. □



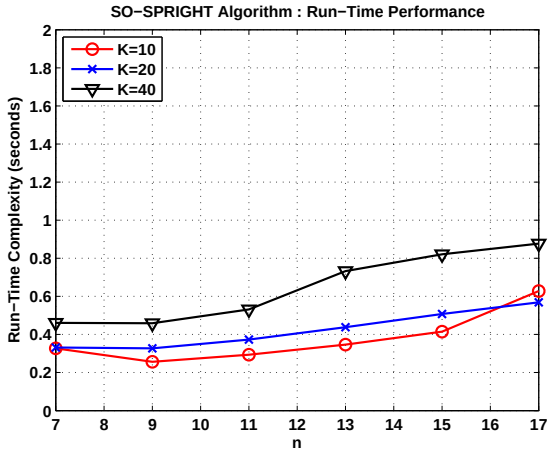
(e) NSO-SPRIGHT : signal length $N = 2^n$ increases by 1000 fold while the sample complexity increases by 5 fold.



(f) SO-SPRIGHT : signal length $N = 2^n$ increases by 1000 fold while the sample complexity increases by 3 fold.



(g) NSO-SPRIGHT : signal length $N = 2^n$ increases by 1000 fold while the run-time increases by at most 6 fold.



(h) SO-SPRIGHT : signal length $N = 2^n$ increases by 1000 fold while the run-time increases by at most 2 fold.

Figure 8: The plot shows the scaling of the sample complexity and run-time of the NSO-SPRIGHT and SO-SPRIGHT algorithms for inputs with varying dimensions $N = 2^n$. With probability of success exceeding 0.95 and sparsity $K = 10, 20, 40$ at a constant SNR of 10 dB, both the sample complexity and the run-time of the NSO-SPRIGHT and SO-SPRIGHT algorithms scale sub-linearly in N (i.e. linear in n^2).